MASTER OF SCIENCE BY RESEARCH

Games as a vector for behaviour change

Richards, Michael P.

Award date:
2011

Awarding institution:
Coventry University

Games as a vector for behaviour change in young children

Michael P. Richards

In partial fulfilment of the requirements for Masters by Research degree

October 2011
Coventry University

Acknowledgements

The completion of this project would not have been possible without several individuals, who provided inspiration, guidance and both professional and personal support.

First and foremost, my supervisor Dr. James Shuttleworth, for his encouragement and support from start to finish.

Dr. Sarah Mount and Dr. Sean Neill for agreeing to examine my work and conduct my Viva.

Dr. Eike Anderson and Dr. John Halloran for their support and advice on preparing for my Viva.

Mrs. Nicola Smith for the help and knowledge provided in the initial stages of the project, and without whom the final product would be lacking in real world application.

And finally to my family and friends, who's continual support and understanding will never be forgotten.

Abstract

It's a common assumption that games have a real and measurable effect on their players. This has prompted the development of numerous educational game titles, but little research has been conducted on their educational impact.

In comparison, a large body of research exists on learning theory and educational practise.

In this thesis a number of parallels are drawn between learning theory and game characteristics in order to determine if it is feasible to accentuate the characteristics of games that correspond in learning theory to facilitate effective learning in games and the manner in which it could be achieved. Drazah is an exemplar of how a game can be designed to enhance personal safety knowledge in an endeavour to reinforce positive behaviour change.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the UK, 800,000 Children aged 0-18 were admitted to A&E due to injuries caused by accidents in the year 2002, 2,760 of which ended in fatality in children aged 0-14 (RoSPA 2002).

Several government agencies and quangos have a remit that includes informing and educating the public in order to reduce the incidence of childhood death and injury. Each of these bodies have different methods and initiatives. For example, the Child Safety and Education Coalition (CSEC) and the Royal Society for the Prevention of Accidents (RoSPA) attempt to reduce injuries caused in home and leisure environments by educating children on hazards and how to avoid them.

The effectiveness of the campaigns run by bodies such as CSEC and RoSPA is difficult to quantify due to a lack of formal evaluation. There are many confounding factors that make such an evaluation difficult. The multiple and overlapping initiatives implemented by these organisations is one example of such difficulties, another is that the timescales needed to effectively evaluate them is longer than they typically run (1-2 years).

Due to the rise in popularity of modern day video games (2.2), young chil-

dren have an increased exposure to this entertainment platform, because of this, games present a possible method that CSEC and RoSPA could use to expand the influence of their educational campaigns.

In collaboration with CSEC this project sets out to determine in what way learning theory and the motivational aspects of games could be used in educational game design. The findings are demonstrated through the development of a fully functional exemplar called “Drazah” that was built incorporating the ideas discussed in this thesis.

The research and technical development was driven by the author with guidance and data provided by CSEC.

This project begins with a summary of the pertinent learning theories deployed in educational game design, as well as some of the characteristics of games that make them enjoyable and engaging and the relationship between the two. These learning theories and game aspects form the basis of recommendations for developing educational games and are later used in the development of Drazah.

This project’s contribution to knowledge is Drazah, an educational card game developed as an exemplar of how numerous aspects of learning theory and motivation can be integrated into a game, in order to influence the player or game in some way.

Chapter 2 of this thesis summarises the background literature considered fundamental to the completion of this project and serves as a brief introduction to topics such as safety education in the UK and the games industry.

Chapter 3 reviews the relevant literature on the psychological and physiological effects games have on their players.

Chapter 4 of this thesis discusses the techniques used in the development

of educational games and how they impact the learning process. This then forms the basis of the framework design presented in Chapter 6.

Chapter 5 examines methods for evaluation of educational tools in general, and derives the requirements for a successful evaluation of Drazah.

Although user evaluation is outside the scope of this project, the author feels it is important to consider the issues of evaluation of educational games, especially in context. In particular, the lack of validated approaches to applying learning theory to educational games is instructive.

Chapter 6 applies the theories discussed in Chapter 4 to the development of Drazah, and demonstrates how the use of experiential learning and Social Cognitive Theory (SCT) had a direct impact on its design.

This project's contribution to knowledge is Drazah (Chapters 4, 6 and 7), an educational card game developed as an exemplar of how numerous aspects of learning theory and motivation can be integrated into a game, and what influence they have on either the player or game design.

1.1 Aims and Objectives

The aim of this project is to determine if games have the potential to facilitate learning, and what can be done to increase the likelihood of this happening. The overall objective of the project is to use this knowledge to design a game to teach children about hazardous situations and objects. The development of the game will be based on the findings of a literature review on approaches to educational game design and common game mechanics. It is theorised that some of the features of games that make them engaging and enjoyable could benefit an educational game.

Chapter 2

Background

This chapter presents the key findings influencing the design of Drazah. These include an overview of the games industry, the potential for games as educational tools, current safety campaigns in the UK and stochastically designed games. It is intended that this chapter provides an introduction into some of the concepts dealt with more in more detail later on in the work.

2.1 Safety education in the UK

There are numerous agencies currently at work within the UK trying to educate children about topics such as personal safety, hazards and the prevention of injury. During this project, support was received by two such agencies, (RoSPA)and (CSEC). Currently these agencies are delivering campaigns aiming to educate children on road safety, electric gate safety, lighter evening awareness, young drivers and public health.

These agencies have seen positive results in their work on the develop-

ment and implementation of new safety regulations, such as limiting house water temperature using Thermostatic Mixing Values (TMVs) in 2010.

Assessing the effectiveness of these campaigns is problematic. The task requires the measurement of long-term effects of short and variable interactions. There are many confounding factors and influences, such as world events, popular school campaigns and so on. These confounding issues cannot be directly measured and thus their impact cannot be compensated for in order to calculate the exact impact these campaigns have on children.

However, through the observation of long term injury statistics, RoSPA aim to find an inverse correlation between the number of injuries and the intensity of safety education.

Game-based learning is one avenue that these agencies would like to explore.

2.2 The games industry

The earliest computer games were created by academic computer science departments in the 1960's and have flourished alongside hardware and software development for the last 50 years. The first ever computer game, 'Spacewar', was created for the PDP-1 computer at the Massachusetts Institute of Technology (MIT) in 1961 by a graduate student named Steve Russel (Herz 2001). Spacewar provided a way of learning the computational capabilities of the new computer technology. Since the development of Spacewar, the games industry has expanded at rapid speed and driven the development of new technologies such as high-grade graphics processors and Artificial Intelligence algorithms (AI), and generated a wide range of game genres and interaction techniques. This expansion has resulted

in a growth of the games industry with annual profits around £400 million in the United Kingdom alone (Economics 2008).

The industry has provided a new option for leisure activities and one that has been taken by many. Research suggest that children and adults alike are becoming more habitual in their gaming tendencies (Prensky 2003).

Figures show that over 26 million people in the UK are playing games, of which 48% are female (Economics 2008). The underlying reasons for the popularity of digital games is explored in further detail in Chapter 3 on page 25.

2.3 Educational games

Non-surprisingly, the popularity of games has led to research on how they can be used in other areas. One of the areas in which a considerable amount of research has been conducted is that of enhancing education using games.

In order to fully understand educational games, we need to examine not only learning theory and game design techniques, but also motivation and engagement in games. An examination of different pedagogical approaches and how they can be applied to games is presented by Kebritchi & Hirumi (2008). By evaluating a game's ability to educate, this work serves as a starting point for understanding how games can be used to deliver educational material and is examined further in Chapter 4 on page 41.

Research has also been conducted on the use of games as a method of learning, including that of Paras & Bizzocchi (2005), Squire (2005), Gros (2007), Kiili (2004), Rosas *et al.* (2002). Computer Supported Collaborative Learning (CSCL) is a theory put forward by Zea *et al.* (2009) in which

an educational game designed to teach children vowels is presented and evaluated. In addition to this, a large body of research has been conducted on the effectiveness of games as educational tools, the key findings of which are studied in Chapter 4.

2.4 Stochastic design

A stochastically designed game relies on the use of data to determine elements of play such as chance, scores and game mechanics. In his work, Shapley (1953) considers the use of stochastic design as a way to determine transitions between potential player positions and how this data can be used to discover an optimal strategy for play. Utilising stochastic design, Shapley states that the length of a game can be predicted by calculating the probability of a final move with regards to the finite choices available to each player. In addition, Shapley searches for the existence of an optimal strategy, based on the reward and cost of each move within the game.

Shapley's work demonstrates the mechanics of stochastically designed games, the influence of which can be seen in alternative turn based state games such as tic-tac-toe, chess and some card games. Although relevant, the work was conducted before the creation of digital games. Due to the increased computational ability of modern digital gaming platforms, stochastic games are able to utilise data in different ways.

In 2010, Electronic Arts (EA) released their latest football game called 'FIFA 11', in addition to game engine updates, updated teams and players EA introduced a live stats feature. By utilising stochastic design, EA were able to integrate an extensive database containing player attributes. This was then linked to the character representation of that real world player (for example, if a player in real life is injured, their game character would

also become injured and unplayable). This intuitive use of real world data is an extremely good example of how stochastic design can be used to effect a game, and add to the interactivity and immersion that games can provide.

A stochastic approach is also used in many popular Role Playing Games (RPGs) such as Fallout 3 created by Bethesda studios in 2008. Like many RPGs, Fallout 3 is loosely based on the d20 system developed for Dungeons and Dragons by a group called 'Wizards of the Coast'. This system is based on assigning numeric values to the following character attributes:

- Strength - *physical strength*
- Dexterity - *agility and speed*
- Constitution - *ability to resist damage and disease*
- Intelligence - *mental acuity*
- Wisdom - *intuition and sense of things around himself*
- Charisma - *force of personality and physical attractiveness*

Throughout the game players will receive additional points to assign to these attributes, often as a reward for completing goals. This design allows a certain level of customisation and players can tailor the strengths and weaknesses of their character to their style of play. An example of how these attributes can be presented to the player in an intuitive way can be seen in Figure 2.4.

The use of stochastic design is employed in this project to create a dynamic scoring system, intended to influence behavioural learning through positive/negative reinforcement. To do this, statistical data from the Home Accident Surveillance Survey and Leisure Accident Surveillance Survey (HASS/LASS) was analysed (RoSPA 2002). Further details on the results of this analysis and the development of the scoring system can be seen in

Figure 2.1: Player Attributes in Fallout 3

Chapter 7 on page 87.

2.5 Key studies

Studies examining the potential for digital game based learning (Prensky 2001*a*) are fundamental to this project and are the basis for much educational game research. The importance of Prensky's work can be seen in the research conducted by Wiklund (2005) and Kiili (2004) who both refer to Prensky's work in positive light. In addition to studying the potential for games as learning mechanisms, how learning theories apply to game design is also considered. The work of Kebritchi & Hirumi (2008) highlights the impact of different pedagogies when used in game based learning, the findings of which are highlighted in more detail in Chapter 3.

Research of educational game theory and learning theory provided valuable information that influenced the design of Drazah. Motivation became a larger factor for a game's success than initially expected, and the work on classification of motivation types by Ryan & Deci (2000) provided key information on how to engage a player. Further analysis of this work can be found in Chapter 3.4 on page 33.

One of the more contentious and highly discussed arguments in the area of video games and behavioural studies is that of determining if violent video games (and video games in general) have a detrimental effect on the player. Although the resolution of this is beyond the scope of this project, the effects of games on players is fundamental to the primary research question. With this in mind, the impartial literature review of Griffiths (1999) provides an insight in to the findings on the fifteen year dispute of the effect of violent video games on children. In addition, an evaluation of methodologies is conducted with the view of determining their effectiveness and the validity of their findings. Further examination of the impact games have on players is carried out in Chapter 3, the issue of violent video games, and the work of Griffiths (1999) is examined more closely on page 38.

Chapter 3

The psychological and physiological effects of games

The aim of this chapter is to take a closer look at the impact games have on players. Starting with the psychological effects of playing games and continuing on to the side effects of long periods of exposure to games. Research is also conducted on the causes of these effects. In particular, the mechanisms games use to motivate and engage players is analysed with regards to how this can be achieved in Drazah. In addition, a study of the research on violence in video games is conducted in order to determine the risk for games to encourage violent tendencies.

3.1 Flow

'Flow', also refereed to as 'optimal experience' describes the state of being entirely focused on an activity, losing a sense of self and becoming intrinsically motivated to overcoming challenge. The work of Nakamura & Csikszentmihalyi (1990) is the culmination of their decades of work on discov-

ering the reasons why people carry out the activities they enjoy. Conducting numerous interviews with chess players, rock climbers and dancers, Nakamura & Csikszentmihalyi looked for any reoccurring attributes that may contribute to why these people do these activities that seem to have no reward other than the enjoyment of the thing itself. The studies showed that test subjects commonly referred to the enjoyment of the activity and the challenge involved in doing it. The recurrence of these two themes suggests that a person is motivated to complete a task when that task presents the correct level of challenge.

'Staying in flow requires that attention be held by this limited stimulus field (the part of consciousness that handles what you are able to focus on at any given time). Apathy, boredom and anxiety, like flow, are largely functions of how attention is being structured at a given time.' (Nakamura & Csikszentmihalyi 1990). Whilst studying flow, it was discovered that when the challenge of a task correlates to the level of a persons ability they become focused on the activity itself. The characteristics of this focused attention, as described by the subject/s, were a merging of action and awareness, lack of worry about failure, self consciousness disappears and a sense of time disruption.

In order to validate their findings, Nakamura & Csikszentmihalyi followed up their preliminary research with situated experience sampling. Rather than relying solely on subjects remembering an experience, situation sampling was used to gather more data on how an activity may facilitate the flow experience. *'Situation sampling refers to that act of random or systematic selection of situations in which observations are to be made with the goal of representativeness across circumstances, locations, and conditions.'* (Shaughnessy *et al.* 2000)

This second level of testing helps to validate the work of Nakamura & Csikszentmihalyi (1990) by reducing inaccuracies caused from asking sub-

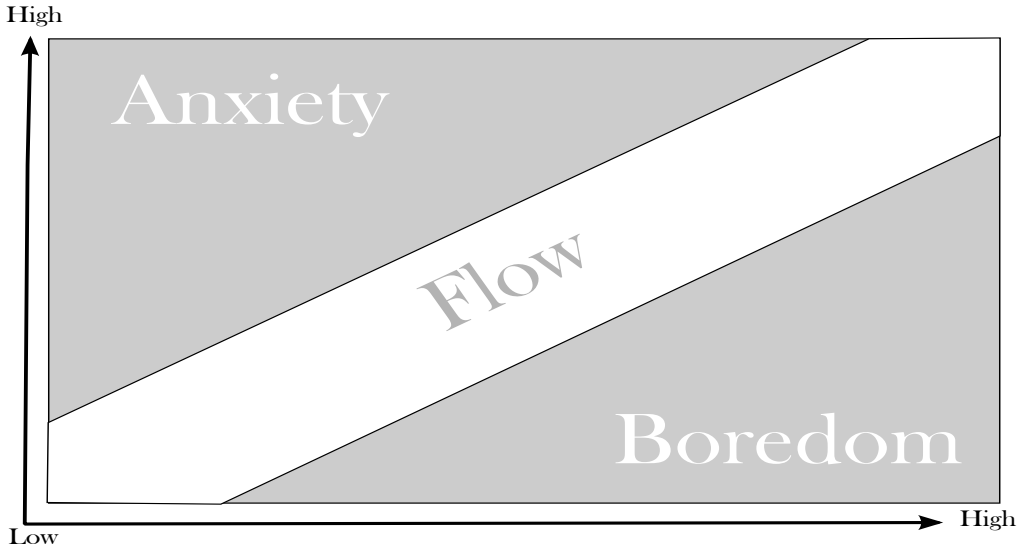

Figure 3.1: Diagram representing optimal experience ('flow')

jects to recall an experience. Relying purely on interviews and questionnaires can provide inaccuracies due to the fact that they rely on retrospective reconstruction, and generally ask participants to summarise an experience that is actually comprised of numerous small experiences (Nakamura & Csikszentmihalyi 1990). The study demonstrates that the level in which a person is engaged with an activity will determine their overall experience. *'What to pay attention to, how intensely and for how long are choices that will determine the content of consciousness, and therefore the experiential information available to the organism.'* (Nakamura & Csikszentmihalyi 1990).

A review of children's media released between 2000-2002 highlighted the impact games have on children and states that *'The final measure of success (with regards to games) is how much fun the game is to play and how effectively the game executes the levels of difficulty'* (Just Kid Inc. 2002).

As we know from the work of Nakamura & Csikszentmihalyi (1990) when a person experiences flow, it is often referred to as being ‘fun’. When coupled with the work of Just Kid Inc. (2002) it can be assumed that games, in some form foster the flow experience in those that enjoy playing them.

The work of Sweetser & Wyeth (2005) extends the theory of flow discussed in (Nakamura & Csikszentmihalyi 1990) and investigates how effectively it can be applied to video games as a method of assessing player enjoyment. By utilising the eight elements of flow as defined in (Nakamura & Csikszentmihalyi 1990) and game playability heuristics, Sweetser & Wyeth combine elements of both to create what they feel is a more accurate method of evaluating a players enjoyment referred to as ‘GameFlow’.

Due to a game’s ability to harbour challenge, present dynamic feedback and foster a player’s attention for long periods of time, it is widely accepted that well designed games can facilitate flow ((Sweetser & Wyeth 2005), (Fu, Su & Yu 2009), (Paras & Bizzocchi 2005), (Kiili 2004), (Squire 2003), (Bellotti, Berta, Gloria & Primavera 2009)). However it is also noted that this optimal experience is not easy to achieve, as stated by Sweetser & Wyeth (2005) in their work on how best to develop games to ensure they engage the player as much as possible.

A comparative study was conducted by Sweetser & Wyeth (2005) on different commercial games was conducted to determine if a significant difference could be identified in how the games were designed to facilitate flow. The study states that by designing for the eight aspects of flow (see appendix) a game is more likely to be considered fun by its player/s. Although this study has had some success in measuring a players enjoyment through games, the testing phases only consisted of a comparison between two games, produced by different companies and rated by different industry experts. The impact of sample sizes on the validity of test results is further examined in Chapter 5. Despite methodology weaknesses, the

ability to expertly look at the design of games using heuristics and determine how these could facilitate flow, in conjunction with the work of Paras & Bizzocchi (2005) help to reinforce the theory that well designed games engage a player in flow.

Paras & Bizzocchi (2005) study how games can utilise flow in order to facilitate learning, and how game designers can help to ensure this optimal experience by designing challenging games. Through examining common traits of games in general and educational games in particular, including challenge, feedback and immersion a correlation is identified between these aspects and the requirements to facilitate flow in an activity, as mentioned by Nakamura & Csikszentmihalyi (1990).

3.2 Feedback loops

'Perhaps the most common usage of the term [feedback] in management psychology and human factors research relates to the presentation of information to individuals regarding different aspects of performance, such as behaviour, strategies, or outcomes' (Atkins *et al.*, 2002).

As stated in the 8 aspects of flow (Nakamura & Csikszentmihalyi 1990), and the work on how this applies to games and enjoyment (Sweetser & Wyeth 2005) one of the features of games that make them enjoyable is their ability to provide the player with feedback toward completing a goal. Further research has been conducted on the importance of providing feedback in games and how this can be achieved by Kiili (2004).

Examination of educational games, specifically the mechanics they possess to help enhance learning, is conducted in order to create a model that *'Stresses the importance of providing the player with immediate feedback, clear goals and challenges that are matched to his/her skill level.'*

(Kiili 2004). When discussing the potential for learning in games, Kiili proposes the use of an experiential learning model; in which reflection is a significant part. One of the ways in which they can provide reflection is through feedback, used to positively or negatively reinforce a players actions and help them stay engaged and challenged. In games, feedback can often be seen in the form of rewards for the completion of goals in order to keep the player informed of their progress.

Feedback loops often comprise of a *'circular motion whereby A effects B, and in turn B effects A again'* Atkins *et al.*(2002). By looking at the method of feedback it is possible to determine its impact on dynamic decision making. Feedback can be used to provide either positive gains, which can destabilise a system (e.g. an in-game achievement) or negative gains which generally regain a state of equilibrium (Atkins, Wood & Rutgers 2002). The work of Atkins *et al.*(2002) contains an examination of the effects different types of feedback delivery have on the receivers ability to conduct dynamic decision making. To do this, an experiment was carried out with eighteen participants using a stock management application, divided in to two groups, each receiving a different form of feedback (tabular or graphical & tabular).

The study showed no variation in response time between the groups, but did depict a significant difference between the quality of performance from each group. The results showed that those who received feedback in a graphical form were able to complete their tasks to a higher standard. Atkins believes that the use of graphical feedback facilitates performance by making the effects of positive/negative feedback more salient, thereby allowing the test subject to focus on the task at hand (Atkins et al. 2002).

The impact of feedback and reflective thinking on facilitating learning in games is discussed in Chapter 4. The work of Atkins et al. (2002) demonstrates the positive impact of graphical feedback on task performance, and

is considered in the development of drazah.

3.3 Neuroplasticity

The topology of the connections within the human brain are not permanent. Repeated activities that stimulate brain synapses can alter the structure of the brain (Draganski *et al.* 2004). This process, called neuroplasticity, is responsible for the learning and mastery of new skills through practise.

Since games at some level are generally built of repetitive activities, it is not surprising that they can influence brain structure through neuroplasticity. Studies, such as the one conducted by Draganski *et al.* (2004) aim to observe this anatomical change utilising Magnetic-Resonance Imaging (MRI). The use of this technology in studies has helped to produce evidence that reinforces the theory that repetition has a physical impact on the brain.

Prensky's work on digital game-based learning introduces the concept of fast-paced data absorption through media such as games and fast image television (up to 100 images a minute). In his work, Prensky goes on to discuss how an immersion of technology from birth has caused young generations (which he refers to as 'digital natives') to become more accustomed to quick data delivery (Prensky 2001*a*). Prensky goes on to say that, due to the popularity increase of games (as displayed in the industry growth statistics by Economics (2008)), children have an increased exposure to this form of data delivery and this repeated activity could be causing a physical change in the way in which children's brains work due to neuroplasticity. Prensky goes on to discuss the difference between this form of data delivery and the traditional classroom paradigm, and suggests

children may become bored in class because they are more accustomed to a faster, interactive form of knowledge transfer.

Practise is required for physical changes to occur in the brain due to neuroplasticity. This is done through continual use of synapses that are required for these actions or thoughts, It is suggested that modern day games provide the levels of repetition required to have this effect: *'A key finding of brain plasticity research is that brains do not reorganise causally, easily or arbitrarily.'* (Prensky 2001*b*). The work of Wiklund (2005) contains a study of behaviour change among game players through neuroplasticity. Although the work discovers some behaviour change in subjects, it is related to which games they played and what other leisure activities they spent their time on. Results showed that test subjects switched their preferred game genre to a more sociable form such as a Massively Multiplayer On-line Role Playing Game (MMORPG) due to the player wanting greater depth from their experience (such as social interaction and customisation)(Wiklund 2005).

Although the work of Wiklund (2005) highlights some interesting points on the ability for games to change a player's brain, the methodology of their experiment is weak. Choosing only male participants, from the same social peer-group and allowing them to pick the games which they play during the study creates an abundance of confounding variables that were not accounted for. In addition, the findings themselves show a shift in preferred game to a more sociable genre, which could be due to the pressures of fitting in with other children in class, among other variables.

It is generally accepted that games have some effect on the player, with regards to neuroplasticity, the work of this section has highlighted some of the key studies and their findings. Prensky's work on digital natives suggests a trend in the way in which children are accustomed to processing data, the impact of which requires further study. The study of change in

grey matter provides strong evidence of the impact repetition has on the brain, however no such results are available to determine the neuroplastical effects games have on the brain.

The effects of neuroplasticity suggest that repetition of an action or behaviour can have long term effects on the anatomy of the brain. However, plasticity research states that this change enables the person to carry out the required thought process more efficiently. This can be seen in the grey matter of people learning to juggle, becoming more proficient in juggling meant a physical change in the brain.

The physical effects of repetition indicate that information can be learnt through practise, and that through enough practise this information can become more efficiently utilised by the brain. The benefits of neuroplasticity for knowledge acquisition and mastery are taken into account in the design of Drazah and have a fundamental impact on the game play mechanics, as mentioned in Chapter 7.

3.4 Motivation

'Orientation of motivation concerns the underlying attitudes and goals that give rise to action - that is, it concerns the why of actions.' (Ryan & Deci 2000).

Motivation is generally considered as the force that drives people to achieve goals, and is divided into intrinsic and extrinsic forms. The work of Ryan & Deci (2000) refers to the classical definition for both intrinsic and extrinsic motivation and how they apply to new research and theory. Intrinsic motivation suggests that a person does something because it is inherently interesting or enjoyable, whereas extrinsic motivation relates to a task being completed in order to gain some form of external reward (Ryan

& Deci 2000).

Intrinsically motivating tasks can vary in form, but are carried out because the person has some interest in doing the task and belief in their ability to complete it. Any form of reward or punishment will undermine a subject's willingness to complete the task for their own pleasure, and the presence of this exterior factor enforces extrinsic motivation. A person can show resentment whilst carrying out an extrinsically motivating activity that is externally propelled, but willingness if the reward is self-endorsed (e.g. 'If I finish this work I can play for 10 minutes'). Ryan & Deci (2000) go on to highlight how positive performance feedback can enhance intrinsic motivation, whereas negative performance feedback can diminish it. Additionally the desire to be socially accepted is stated as the primary cause of extrinsic motivation, generally people are likely to perform a task if it is important to a significant other (e.g. family member, peer group or society).

Using the classical definitions of motivation mentioned above, combined with what can be seen about engagement and enjoyment of playing games in section 3.1 it is possible to determine how games may motivate players. Studying the motivational aspects of games is significant for the design of an effective educational game. It is important that the game engages and motivates the player into interacting with the educational material included. Although the work of Sweetser & Wyeth (2005) (mentioned in section 3.1) examines enjoyment in games and focuses on the evaluation of enjoyment through observational analysis of flow, the continued work of Ryan *et al.* focuses on the motivational effects of video games, and is therefore more suited to studying which aspects of games enhance/diminish motivation.

In psychology, Self-Determinate Theory (SDT) is used to encapsulate a person's growth tendencies and innate psychological needs. Essentially it focuses on the degree to which behavior is self-motivated and self-determined. Applying SDT in their studies, Ryan *et al.* were able to deter-

mine if games provide motivation and if so, which features do so and what form of motivation they provide. Through observational analysis, the study showed that whilst playing games, test subjects would display signs of in-game autonomy and competency. That is, the player had mastered the game environment which allowed them to carry out tasks with ease, and displayed confidence in the understanding of their goals and their ability to complete them. In addition, competence and autonomy were also applied to the physical manipulations of the game's controls and the sense of immersion during play. Ryan *et al.* (2006) state SDT's theorised needs for autonomy, competence and relatedness independently predict enjoyment and game replayability.

A motivational technique deployed in modern day games is the use of rewards, which can be seen by the inclusion of game achievements in every Microsoft Xbox 360 Game to date (Brodzki n.d). According to the work of Ryan *et al.* any form of external reward will result in extrinsic motivation; this in theory could reduce the subject's willingness to complete the task. The work conducted by Brodzki (n.d) states that the acquisition of achievements is often motivated by social factors, including the desire to appear competent. We know that games are considered to be fun, and that features such as rewards enforce extrinsic motivation. However the use of feedback to reinforce behaviours, enhances the player's sense of competency and thus encourages intrinsic motivation (according to SDT Ryan *et al.* (2006)).

The idea of a task providing two juxtaposed forces of motivation is inherently interesting to educational design. On one hand, games provide intrinsic motivation through ease-of-use, immersion, feedback and engaging the player in flow. In contrast, the use of rewards such as achievements or additional game features (tools, clothes etc) enhances extrinsic motivation. The integration of social functions such as player chat and leader boards increases the likelihood that a player would be externally motivated

to complete a task.

The report conducted by Brodzki (n.d) discusses the motivations behind collecting achievements, and looks at the behavioural patterns of pathological achievement hunters. Achievements present an interesting facet of motivation in games, when presented as a reward for completion of a task they are defined as extrinsically motivating. However if a player sets out to collect achievements for no other reason than the enjoyment of challenge, then they are defined as intrinsically motivating. The aims of achievement hunters however, is to collect easily obtainable achievements in order to raise ones social ranking, thus reinforcing extrinsic motivation.

Ryan *et al.* state that due to the wide variety of game genres, the most practical models for motivation are the traditional ones that apply to motivation in general. Applying SDT to game evaluation assisted in highlighting the importance of autonomy, competence and relatedness with regards to the design of Drazah. This relates well with the findings in section 3.1 on how flow can be established to enhance enjoyment. The details of how this impacted the design of Drazah can be seen in Chapter 7.

The impact of social aspects on motivation, as mentioned by Ryan *et al.* (2006) is increased dramatically when considering the intricate ways in which social interaction has evolved in games. The use of a globally visible rating system can motivate players to achieve goals purely to gain social capital, as seen in Broski's report on achievement hunters in games. In order to determine if educational games could benefit from players being extrinsically motivated through social aspects, a closer examination is made on the impact of social capital in modern games.

3.5 Social capital

Due to the limited computational and networking abilities of gaming hardware in the late 60's/early 70's, games were mostly single player, which created the stigma of game playing being an unsociable activity. Since then, however, through the introduction of vast networks (including the Internet) that perception has changed (Harris 2001). Development of network capabilities resulted in a rise in the creation of multi-player games (both cooperative and PvP) as well as the introduction of social media such as chat functions and friends lists as well as player comparison such as leader boards.

Evidence of this evolution in pro-social gaming can be seen in the development of both Sony and Microsoft's gaming communities (PSN & Xbox Live), player ranking systems and achievements. Valve's gaming platform 'Steam' also contains community pages, forums and the functionality to create profiles and groups of friends.

Although the social aspects of games are not gender specific, Baranowski *et al.* (2008) and Gros (2007) both state that in play, girls are often more interested in the social aspects of games and realism, while boys are more interested in action and adventure. A study conducted by Herz (2001) looks at the formation of in-game ecosystems, in which players contribute to an expanding society and knowledge base in an attempt to gain social acceptance. The use of a social capital, such as a ranking system, achievements or player statistics can lead to players being extrinsically motivated in performing tasks that will influence their position within the gaming community. Herz goes on to state that this self-sufficient form of motivation is beneficial to education, as gamers constantly wish to increase their social standing through personal development.

Several features designed to serve as social capital have been developed

for Drazah, based on the research discussed in Chapter 3 both achievements and player statistics are included in the game. These are designed to encourage motivation through social capital and self determination, and thus increase the likelihood of flow being established. Further details on the development of these features can be found in Chapter 7 on page 87.

3.6 The impact of violent video games

When designing educational games it is important that the potential for behavioural change in players is considered, therefore the results of a study on the physical and psychological effects of games, carried out by Griffiths (1999), have been discussed below.

Arguably the most controversial topic for debate on the effects of video games is the impact themes of violence have on young players. One of the issues in this debate is that there is no standardised definition of violence with regards to game content or themes (Griffiths 1999). Griffiths highlights the implications of this lack in definition and states that this could be one of the reasons why there is so much controversy on the subject. Griffiths (1999) proposes that this lack of clarity, in addition to design approach, social context and research methodology have contributed to the number of studies that produce inconclusive results.

Common methodologies in this field include self-report, examination study and observational study approaches. Potential issues with these approaches include lack of pre-test evaluation for comparison, irrelevant post-exposure testing and confounding test variables (Griffiths 1999). In some cases variables such as educational attainment and socioeconomic status were considered to be attributing factors to test results.

In an attempt to produce empirical evidence, one study examined the

physical effects of games between players of violent and non-violent video games. It was hypothesised that playing games with violent content would increase cardiovascular responses, this was examined by monitoring heart rate & blood pressure during and after play (76 subjects). The results of this study showed no significant deviation between the physical attributes of the two groups and adds to the research suggesting violent games do not have a physical effect on players.

In addition to evaluating methodologies and analysing results, Griffiths (1999) also takes a theoretical approach to the field of study in an attempt to find a reason for the vast numbers of contradictions. A possible cause of this contradiction, is that theoretically games have the potential to either increase or reduce violent behaviour. In accordance with social learning theory, videogames have the potential to promote aggressive tendencies. In contradiction, catharsis theory suggests games provide an avenue in which the player can release their violent tendencies in a safe manner, thereby reducing their aggressive behaviour (Griffiths 1999).

Griffiths (1999) summarises their work by stating that a generalised view within the field exists regarding the need for refinement and validation of testing methods. And that in accordance with social learning theory and catharsis theory, games do have a behavioural impact on players; however studies lack any empirical evidence to determine if this effect is positive or negative, or if it is significant or even measurable.

Chapter 4

Evidence for the efficacy and effectiveness of educational games

This chapter of the thesis carries out an evaluation on the potential to use games as educational tools, reviewing literature on the efficacy of educational games and demonstrating how learning can be made more effective through games. The key findings of this chapter determine the potential for using Drazah as a behavioural education tool and have a direct impact on its design.

4.1 An examination of approaches to educational game design and methodology

'Digital game-based learning is precisely about fun and engagement, and the coming together of serious learning and interactive entertainment into

a newly emerging and highly exciting medium' (Prensky 2001*a*).

In his book entitled 'Digital game-based learning'' Prensky looks at the potential for games as educational tools and suggests the physical changes that may be occurring within a generation of consistent game players. Due to the development of fast paced media such as computer games and fast image television (60 > images per minute), the way in which young children are accustomed to receiving information has changed. Prensky refers to this as 'twitch speed' and highlights the stark contrast between this form of data delivery and that of a normal classroom paradigm. *'The reason most kids don't like school is not that the work is too hard, but that it is utterly boring'*(Prensky 2001*a*). This technological change may have a significant physical impact, and in an attempt to prove this Prensky looks at the change in young children's brains due to the effects of neuroplasticity. In an attempt to increase motivation in schools, Prensky discusses the potential for games as educational tools and explores the motivational and engaging benefits of game-based learning.

Expanding on the principles of game-based learning, the works of Amory *et al.* and Squire (2003) look at the development processes of playing games and the history of games in education respectively. Both pieces of work state that games have the ability to educate, and examine the advantages and disadvantages of digital game-based learning. The work of Amory *et al.* (1999) starts by studying at the motivational rewards of play and how this effects self discovery and learning. In an attempt to harness the motivational side-effects of video games, Amory *et al.* conduct player evaluations of four different game types (Strategy, 'Shoot-em-up', Simulation and Adventure). A group of twenty students were used and each were required to complete a questionnaire after playing each game for one hour.

The aim of this evaluation was to determine what type of game was the

most popular with students, and determine which game characteristics were most sought after by the players. Results showed that the most popular game genre was adventure, closely followed by strategy. When questioned about their choices, students stated they enjoyed the challenging aspects of the games and found them engaging. These responses are what we would expect to find from players experiencing flow whilst playing, and correlates with research on user engagement and enjoyment (see Section 3.1 on page 25).

Complementary to the work of Amory *et al.* on popular game types, the research conducted by Squire (2003) looks at the elements of video games that make them engaging and fun. The study highlights the importance of clear goals and feedback, scoring, challenge, randomness and emotionally appealing fantasy within games. Research suggests that despite the rapid advances in recreational game technologies, little has been done in the form of educational game development except for commercially available educational games known as 'edutainment' products. In some cases these products are used in classrooms without any empirical research in to their effectiveness at teaching and their impact on classroom dynamics (Squire 2003).

'To achieve students' attention, new ways to utilise computers must be found. The use of technology alone does not motivate students that have lived in the midst of technology all their lives. Thus, learning situations and methods that engage learners must be created.'(Kiili 2004). In an attempt to do this, Kiili uses flow, described in (Section 3.1) as a framework for creating an integrated model based on educational theory and game design aspects. In addition, the work examines the importance of making an educational game fun by paying attention to game aspects such as visualisation, storyline and game balance. Throughout their research, Kiili stresses the importance of engaging the player in flow in order to engage and motivate, as well as facilitate heightened cognitive functionality. The

study separates educational games into three sections, person, task and artifact, and states that it is preferable if any learning content (referred to as artifacts) are as transparent as possible. If a task or artifact is too complex, then valuable cognitive resources could be taken away from learning. In order to maximise learning, artifacts should appear as natural elements of the game, and designed to be easy to interact with (Kiili 2004).

In an attempt to reduce cognitive load, Drazah is designed to present artifacts as game elements in an abstract way. Hazards and actions are designed as cards to be played in order to score points. The work by Squire (2003) on the engaging elements of games and the advantages of motivation also has an impact on the design of Drazah.

As well as studying elements of game-based learning and flow, significance is placed on the use of an effective educational theory or pedagogy (Kiili 2004). In their work, Kiili focuses on the use of an experiential learning model within educational games and how the aspects of learning theory apply to attributes of games. *'According to the (experiential) model, learning begins with a concrete experience followed by collection of data and reflective observations about that experience.'* (Kiili 2004). During abstract conceptualisation the learner draws conclusions and creates hypotheses, the learner then tests these hypotheses through active experimentation. By reflecting on how learning theory demonstrates the process of knowledge acquisition, Kiili hopes to develop a framework for a game that facilitates this process of thinking and thus enhances learning.

In addition to facilitating academic education, it is widely accepted that games can develop skills such as strategic thinking, communication, application of numbers, group decision making and data handling (Kirriemuir & McFarlane 2004). An examination of literature on educational games discovers that there are two main motivations behind game-based learning research.

1. The desire to harness the motivational power of games in order to 'make learning fun'.
2. A belief that 'learning through doing' in games such as simulations offers a powerful learning tool.

The difference in these approaches tends to determine the researchers main focus. Those with the motivation '1' *(above)* are more likely to focus on the attributes of the game that make it enjoyable and motivating. Whereas those with motivation '2' are more likely to focus on the underlying pedagogies and how best to facilitate effective learning. Evidence of this can be seen in the engagement and motivational approach adopted by Bellotti et al. (2009) and the games created from experiential and constructionist pedagogical foundations studied by Kebritchi & Hirumi (2008).

Kirriemuir & McFarlane (2004) conducted a review on the key themes in educational games including motivation and flow, game genre categorisation, differing approach frameworks, game exposure and video game violence. Although reflective in nature, the literature review aids in highlighting some of the trends in the area of educational game design, and emphasises the importance of flow. The study states in order to take advantage of motivation and flow, emphasis should be placed on making a game enjoyable, and that educational material should be considered at a later stage. In contrast, the work of Kebritchi & Hirumi (2008) analyses the advantages of focusing on a pedagogical foundation within an educational game, and reflects the popularity of this approach in other work. In addition, the work of Conati & Zhao (2004), Zea *et al.* (2009) and Chuang & Chen (2009) all state the importance of including a pedagogical foundation during the design of an educational game.

Examining the advantages of using games as educational tools requires a two-pronged evaluation of game design and learning theory. Chapter 3 highlights the impact games have on players, and as such serves as a

basis for considering the potential for games as educational tools. The literature on the efficacy of games as learning tools (Chapter 5) presents several differing approaches to educational game design. Among these, emphasis is placed on ensuring the game is engaging and fun by facilitating optimal experience, and support effective learning through the use of a pedagogical foundation such as experiential or guided learning theory. These ideas are considered in the approach of designing Drazah, in which game features are designed to encourage motivation, replay-ability and enjoyment. Further details on the development of Drazah's learning approach can be found in Chapter 6.

4.2 Increasing the effectiveness of educational games

A novel approach to increasing the effectiveness of educational properties in games is that adopted by Kafai (2006), in which 'instructionist' and 'constructivist' approaches are considered for game based learning. *'Instructionists, accustomed to thinking in terms of making instructional educational materials, turn to the concept of designing instructional games. Far fewer people have considered making games for learning rather than playing them'* (Kafai 2006). In this article Kafai looks at the benefits of designing an educational game with students as a problem based learning approach to educating children. Assisting groups of ten year olds to design and make a maths games to teach factors to younger children, Kafai observed differences in the approach taken by all students.

Although no statistical evidence was gathered, the study shows high levels of engagement in students and enjoyment and motivation toward learning. In addition, the constructivist approach meant that students had exposure

to the game environments and thus gained an understanding of the learning material involved in the game. Although no difference was found in the motivation and engagement among students of different genders, Kafai does note a difference in game design and themes, including the use of violence as a punishment. The study showed that when making a maths educational game, boys were more likely to use violence as the result of an incorrect answer than girls. It also noted that when changing the subject of the game to science, this gender divide in use of violence dissipated, although Kafai does not state why. Despite the lack of empirical evidence, the novel approach to educational game design is enlightening and emphasis is placed on the importance for the development of educational environments in which a constructivist approach to learning can be held.

In order to determine the effectiveness of educational games and to evaluate their impact on classroom dynamics, Rosas *et al.* (2002) conduct an observational analysis of an educational game. The study focuses on the implementation of two educational games designed to teach mathematics and reading skills. To do this 1,274 test subjects were divided into an experiment group, and two control groups, they were then evaluated for their acquisition of skills through ad-hoc tests and observations. The scale of this evaluation showed real promise with regards to its potential impact on educational game research, and in addition the use of multiple control groups helps to validate any data through comparison. Using Nintendo's 'GameBoy' platform for the basis of this test Rosas *et al.* (2002) note a significant improvement in classroom dynamics, student morale, punctuality and discipline. In addition, a positive but not significant difference was seen in the post-test scores for maths and reading skills of the experiment group over the two control groups (Rosas *et al.* 2002).

Examining the potential for the implementation of games within the classroom has uncovered some positive results, in addition to the work of Rosas, Nussbaum, Cumsille, Marianov, Correa, Lopez, Rodrigeuz & Salinas (2002)

Squire (2005) and Wiklund (2005) both reveal positive responses from students. Despite these findings, only the work of Rosas et al. (2002) boasts a significantly robust methodology worthy of merit. The findings clearly identify the ability of game based learning to deliver academic material effectively, however they do not specify any details of how the game was designed to facilitate learning. In order to understand how games can potentially educate, emphasis is placed on understanding the learning theories and pedagogies used in educational game design.

Experiential learning was the most commonly identified pedagogy in game based learning, followed by discovery learning, situated cognition and constructivist learning (Kebritchi & Hirumi 2008). In their review, Kebritchi & Hirumi examine the pedagogies of 22 educational games in an attempt to discover reoccurring patterns in approach and design. Experiential learning theory was the most used pedagogy and was found in 8 of the 22 games studied. *'During experiential learning, educators purposefully engage learners in direct experience and direct their focus on learning reflection to increase their knowledge, skills and values'*. In an attempt to facilitate experiential learning, some games created a resemblance of the real world in order to help link learning material with real world values. This was done through aspects such as story, characters or objects.

'Learn by doing - Instructional theory is based on learning by doing, and fosters skills through practise.'(Kebritchi & Hirumi 2008). In an attempt to engage players in an instructionist learning method, games often attempt to guide players through knowledge-rich games and encourage them to repeat new skills. This theory relies on exposing a learner to a new concept, and providing a period of reflection in which the player can observe their progression. The use of an instructional approach to educational games is also explored by Conati & Zhao (2004) but does not bare any empirical evidence to suggest its efficacy. Utilising an instructional approach to learning means that a game must have a concept that is intended to be

learnt. Player interaction with this concept can be provided by embedding it in to a context or situation. *'Knowledge is situated in its context, more specifically, knowledge is a product of its context activity and culture within which it is developed and used'*. Situated cognition relies on embedding a topic within its context, and allowing it to be learnt through interaction with that context.

This review on learning theory in games allows for the identification of popular trends and how they are designed to help facilitate learning. In addition to the inclusion of a pedagogy, seventeen of the games mentioned, simulated real world experiences, thus highlighting the potential for representing realism rather than fantasy in games. Kebritchi & Hirumi (2008) state the importance of including a pedagogical foundation in game-based learning and state that it was beneficial to include a pedagogy that *'allowed questioning and active experimentation'*.

Both the work of Conati & Zhao (2004) and Bellotti et al. (2009) are focused on increasing the educational effectiveness of games, but aim to do so in different ways. Acknowledging the importance of feedback and discovery, Conati & Zhao (2004) look at creating an intelligent hint and feedback service to embed within their experimental game. By doing so they aim to increase the experiential learning foundations of their game by facilitating the player in self led discovery and reasoning. As an additional feature this 'intelligent pedagogical agent (referred to as a hint system) helps to better adjust the games challenge to the player's skill levels. This is done by providing suggestions when the player appears to be struggling, or removing possible incorrect answers to help improve the likelihood of the player succeeding. All of these features are designed to help facilitate flow and reinforce the experiential learning theory designed to assist in knowledge acquisition. By evaluating the difference between pre and post-test scores Conati & Zhao (2004) are able to state with some significance that their hint system resulted in positive gains among their experiment group

in comparison to their control group.

The work of Bellotti et al. (2009) concludes this study on effectiveness of educational games by stressing the importance of using the technological advances games have to offer for educational benefits. To demonstrate this, Bellotti et al. create a simulation game in an attempt to teach players new behaviours regarding safety near water (mainly the sea). This is done through immersing the player in a story and goal-based self discovery play style, in which the player is able to interact with marine equipment such as jet skis and boats. The study is based on enhancing existing games for educational effect by implementing sound learning theory, rather than building a game upward from established pedagogical foundations.

4.3 Behavioural education

The focus of this chapter so far has been the effective use of games as education tools through the examination of games designed to deliver skills and/or academic material. Due to the nature of safety education, Drazah is specifically designed to reinforce positive behaviour changes by educating children about hazards. Therefore at this point, it is worth mentioning in detail the results of research in the field of games for behavioural education, including highlights of studies and findings from other research.

Much of the basis for behavioural education can be seen in Social Cognitive Theory (SCT), which looks at the impact of observation on the acquisition of new skills. In addition, SCT states the importance of self-efficacy, meaning the point in which a learner has belief in their ability to accomplish their goal/s. The effectiveness of this educational approach has been widely discussed, and research has been conducted to help quantify its impact on behaviour change. The work of Burke *et al.* (2002) shows re-

sults from several studies of behavioural education among newly married couples with regards to their physical activity and diet. In an attempt to reduce negative behaviours such as high consumption of unhealthy food, or lack of exercise, Burke *et al* (2002) developed a health program designed around goal setting and social support in order to teach new, healthier behaviours. An initial study of 39 couples, and further study of 137 couples (over a longer period) provided evidence to suggest a positive change in subject health and diet.

By measuring physical characteristics such as waist circumference and blood pressure, Burke *et al.* (2002) demonstrated how their self-efficacy and social framework for behavioural education has changed test subject's habits. In addition, by extending the latter testing phase to 12 months, they are able to present a positive retention of these new traits. In conclusion to their work, Burke *et al.* (2002) state that there is potential in using SCT as a framework for behavioural education, although further assessment is required.

The work of Baranowski *et al* (2008) is a review of twenty five games designed as behavioural education tools aimed at facilitating positive health-related behaviour change. Looking at the mechanisms behind behaviour change and evidence of its effectiveness, Baranowski *et al* (2008) positively demonstrate video games as a channel to promote behavioural education. The games studied tackle a series of topics including changing diet, physical activity, physical activity among physically challenged, diet and physical activity combined and positive health related medical issues. Results of this evaluation show a positive response in almost all cases especially among pediatrics and asthma suffers (Baranowski *et al.* , 2008).

The large sample size of games provides Baranowski *et al* (2008) with a vast body of data to help validate the effectiveness of behavioural education in games. In addition, it enables a comparison of game features

to be made, and a theory of best practise is presented. *'A comprehensive model of learning for behaviour change in video games is based on social cognitive theory (SCT) and the elaboration likelihood model, and includes the following steps: attention, retention, production and motivation'* . Broken down, this means that the first step toward behavioural change is gaining and maintaining someones attention (elaboration likelihood). Secondly, the need to demonstrate a new behaviour by clearly highlighting the positive aspects, according to SCT. And finally, allowing the learner to practise these skills and enhance their confidence, engaging the learners self-efficacy so they become more likely to repeat the behaviour until mastery.

The results of this analysis demonstrate the potential for games designed to educate players on new behaviours, and suggests ways in which games are able to do this. The findings on game play and flow (Section 3.1 *'Flow'* on page 25) state that games have the ability to engage their players with heightened attention, thus fulfilling the first stage of the framework proposed by Baranowski *et al* (2008). Once motivated, players will continue to practise new skills in an attempt to complete goals, whilst receiving constant feedback on the level of their ability, thus satisfying SCT.

This chapter has highlighted some of the techniques deployed in current educational games and what impact they had on the players and game play. SCT can be seen to increase a player's engagement of a game by as measured by attention, retention, production and motivation. It is suggested that this increased engagement can temporarily increase cognitive functionality and have a positive effect on learning and knowledge acquisition. Multiple pedagogies were also observed in order to determine what impact they have on learning and game play.

Chapter 5

Evaluating educational games

The development of Drazah is intended as a synthesis of ideas in the field of educational games, based on the findings on design approach and learning theories, highlighted in Chapters 3 & 4. Drazah serves as a demonstration of how these aspects of educational games can be implemented into the design of a behavioural education game. This research based approach was adopted by the author for two reasons:

1. Complications in the testing of educational games has thus-far prevented any unified theory of evaluation to be agreed upon within the field.

2. The time and resources required for the successful evaluation of an educational game are greater than those provided by this project.

This chapter serves as an evaluation of testing methodologies encountered through literature search in order to highlight positive and negative aspects. This is then concluded by stating the requirements for a successful evaluation of an educational game such as Drazah.

5.1 Heuristics

Heuristics are an experience based technique for solving problems, based on the optimum of a mathematical function, and are used to discover an optimal solution. Heuristic approaches such as 'trial and error' can be applied to problems in numerous fields, including Science and Geography, when applied to psychology, heuristics refer to simple alternatives to optimisation models (Katsikopoulos 2010). Within the field of computer interaction and software development, heuristics are used as a guide of best practise, and serve to ensure effective and efficient design.

A widely used approach to the evaluation of effective educational game design, heuristics help determine what aspects of games help engage the player and make the game fun and playable ((Garzotto 2007), (Fu et al. 2009), (Eagle 2009) and Barendregt *et al.* (2003)). The work of Fu et al. (2009) and Garzotto (2007) specifically focus on the development of a framework of evaluation based on system design heuristics. This is then tested on a single game to determine how effectively the framework applies to game design as well as observing any knowledge acquisition. Although effective in determining the usability and design of an educational game, heuristics are primarily used to determine how easy a game is to play. They therefore lack any method of determining a change in the player's knowledge level, on the basis that an educational game is designed to increase the knowledge of the player, a method for determining this is essential for the effective evaluation of an educational game (Eagle 2009).

The work of Barendregt, Bekker & Speerstra (2003) highlights two key issues with using heuristic observation and task setting for game evaluation. Essentially, adopting a heuristic approach to testing requires the test subjects to complete certain tasks to ensure they have exposure to

the predetermined content. For example, in order to test the functionality of a specific menu bar, the test would require the user to use it in some manner. In order to do this with games, tasks are set that potentially disrupt flow and engagement, *'The major problem with giving tasks for the evaluation of computer games is that the goals set by the tasks interfere with the intended goals of the game.'* In addition to this potential break in flow, heuristic evaluation is ultimately decided by an observer of some form. It is the responsibility of the expert evaluator to identify an issue and categorise it within a certain heuristic so that it can be evaluated later. *'There can be a lot of disagreement between different evaluators because of vague evaluation procedures and problem criteria.'*.

The findings of these studies highlight the inefficiency of using heuristic observation for the evaluation of educational games. It is likely that the use of this method was adopted due to its success in evaluating the effectiveness of software such as web services. And considering the engagement levels that games have shown to possess (section 3.1) ease of use is essential to effective game development. *'Ease of use of a game's controls and interface is a pervasive factor that impacts on all the other aspects of the user experience'* (Garzotto 2007). However heuristics lack the ability to evaluate any educational impact of a game and are therefore ineffective as a method of evaluation for educational games.

5.2 Pre-post test

A novel approach to effective game evaluation is put forward by Eagle (2009) in their work on coupling heuristic observation and mapping of student data (pre-post test results) on to game-log data. After stating the need for empirical evidence in order to determine the educational impact of games. It is hypothesised that by recording game play and testing for

knowledge acquisition, it is possible to determine which features of the game provided educational support. This framework is then applied to a test case in which 61 children played a game designed to teach programming concepts, results showed an initial decrease in time and number of attempts taken for the player to progress through the challenges.

The use of pre and post test quizzes provides this study with some empirical evidence to suggest knowledge gains have occurred during the testing phase. However there is little evidence validating that this learning occurred from the game itself, or if it did, how effective this was in comparison to alternative teaching methods. In addition, the effective plotting of data requires extensive observation during testing, Barendregt et al. (2003) states the negative aspects of observation during testing and how it can effect the test results as the players feel as though they are being tested.

The use of pre and post test evaluation allows for some comparison to be made in the level of knowledge before and after playing the game. By doing this, it is possible to determine if the player has gained any additional knowledge during their exposure to the game.

Although the use of statistical evidence is presented in this study as a way of proving educational benefit, there are many issues with the methodology applied that potentially lead to insignificant findings. Issues such as sample size, confounding variables and lack of control group prevent any data validation from being conducted. Although the study looks at the results of 61 test subjects, the evaluation only requires playing one educational game, this means that no comparison can be made regarding how suitable this approach is on a broader range of games. In an attempt to determine the effectiveness of this evaluation framework, a greater sample size of games is required.

5.3 Sampling

One of the reoccurring issues within the realm of video game evaluation is that of confounding variables that are unaccounted for, this is evident in the study conducted by Garzotto (2007) and many others in section 3.6. Due to the dynamic nature of the interaction between player and game, and the various different scenarios in which a person can play a game, there is a vast potential for variables that can affect the impact of games.

The strongest way to state facts is to establish an empirically reinforced statement, consisting of data that indicate the following (McLean & Ernest 1998):

- The findings significance *(provides evidence that an event did not happen by chance)*
- The meaning *(practical significance)*
- If the findings are replicable *(results can be repeated)*

Within educational games it is essential to determine how significant a change in learnt criteria is (if any), state the potential for this form of education and be concise enough in experimentation that results can be replicated.

The use of a control group allows for the value of an event to be directly measured, usually through comparison of non-exposure groups or results of competing events. Within educational game analysis a non-exposure control group may consist of a group that continues learning through their normal classroom paradigm. By doing this, comparison of quantified learning can be cross examined between the two data delivery techniques and the effectiveness of each can be concluded. Comparing results from competing events such as two different educational games would enable cross

comparison of the effectiveness of each. Conclusion can then be drawn on the differences between the two educational games as learning tools. An example of this can be seen in the work of Conati & Zhao (2004) & Rosas et al. (2002) in which both studies compared the effectiveness of their educational games against control groups.

Examining the impact of games on classroom dynamics and learning, Rosas et al. (2002) use an internal and external control group for comparison with their experiment group. The criteria for each of the groups were:

- **Experiment group (EG)** - Used the experimental video game in the classroom, for an average of 30 hours over a 3 month period.
- **Internal control group (IC)** - Attended class as normal, but was in the presence of the EG.
- **External control group (EC)** - Attended class as normal and was not made aware of the use of an experimental game.

By having both Internal and External control groups, the study was able to compare the effectiveness of the educational game against the normal classroom paradigm by comparison with the EC. In addition, the impact of games in the classroom could be examined through the performance results of the EG and IC.

The various aspects of these methodologies highlights the difficulties in accurately assessing the effectiveness of educational games. Control groups were used effectively by Rosas et al. (2002) in evaluating the effectiveness of an experimental game in the classroom (maths and reading comprehension were statistically significant enough to determine gains in knowledge greater than the EC group), and classroom dynamics (EG & IC groups showed improved motivation, behaviour and punctuality to class). The lack of an adequate sampling size in the study conducted by Eagle (2009)

means that the significance of the findings are difficult to validate. By increasing the sample size of educational games, the impact they have on learning can be assessed more thoroughly and the capability of the proposed evaluation framework can be comparatively analysed.

Although no evaluation of Drazah was conducted, research suggests that the following would need to be considered if testing were to be carried out in the future.

1. A large sample size will be required, containing at least one control group depending on where the game will be testing (e.g. classroom or home) and enough individuals to ensure significant variation in results is possible.

2. A method of delivery for identical material will need to be designed for the control group, this ensures comparative analysis can be conducted on the efficacy of each learning method.

3. To ensure behaviours are learnt, evidence of physical changes indicative of learning new skills (such as neuroplasticity) should be observed in the brain, using appropriate technologies (Such as an MRI) (Section 3.3) therefore a long period of testing is required to provide ample repetition.

4. External validity relies on the ability to generalise the findings of an experiment and to do this we must ensure that all unnecessary variables are removed from testing, and if necessary all nuisance variables are compensated for.

Chapter 6

Design of the game

6.1 Introduction

One of the key aims of this project is to determine the most effective ways to accentuate the characteristics of games that correspond to learning theory. Chapter 4 presents learning theories such as experiential learning and social cognitive theory and discusses how these can be incorporated in to examples of educational games. Chapter 6 presents the mapping of these ideas to the game mechanics of Drazah, an educational game developed as an exemplar of the incorporation of learning in games. These theories are chosen for their applicability to educational game design, as discussed in Chapter 4, and are individually examined, details of their use in Drazah are presented in the following sections.

The game being developed for this project is called ‘Drazah’, an educational card game specifically designed to increase the players’ understanding of the likelihood and severity of every day hazards. By incorporating proven learning theory, such as experiential learning (Chapter 4), and implementing motivational aspects of games, such as social capital (Sec-

tion 3.5), it is intended Drazah will produce an engaging learning experience for children.

Chapter 6 begins by determining the educational aims and requirements for Drazah, which will be used to determine the scope of the game as well as highlight essential features for inclusion. For example, the work of Kebritchi & Hirumi (2008) highlights the importance of including a sound pedagogical approach to help facilitate learning, and is a key motivator behind the inclusion of an experiential learning pedagogy in Drazah. It is also important that the game uses features of modern-day games that players come to expect, features such as challenge and achievements.

Section 6.3 develops the framework for Drazah's educational approach by progressively incorporating aspects of experiential learning and Social Cognitive Theory to game features. This process is discussed by presenting each segment of Drazah's learning approach individually, and highlighting how it is mapped to the game design. This is then presented and justified as a complete learning approach at the end of Section 6.3.

The development of all of Drazah's game features, including achievements and scores, are presented in Section 6.4. The impact of implementating these features with regard to learning and/or engagement is also discussed.

The functional requirements of a scoring system, and the motivational benefits of feedback, coupled with the use of real-world data led to the development of a stochastic scoring system. Real-world data was used to enhance the games learning content, and was developed in association with CSEC. Further details of which can be found in Section 6.5.

In order to better determine the impact of experiential learning and social cognitive theory on the design of the game, a prototyping phase was carried out to better understand the way in which the mechanisms of the

game would affect its playability in the most basic sense, and to identify issues that might detract from a smooth experience, such as unnecessary or excessive complexity or breaks in play. The importance of flow (Section 3.1) in games is highlighted in the work of Kiili (2004), in which flow theory is used to facilitate positive user experience and maximise the impact of educational games. The playability evaluation helps to identify issues which might disrupt player engagement, and subsequently decrease the likelihood of flow.

6.2 Educational aims of the exemplar

Drazah is created as a demonstration of how learning theory can be incorporated in to the design of a games mechanics and features. The work presented in Chapter 4 is used to outline the educational aims of the game.

The advantages of embedding a proven educational approach within a game and tailoring game features to help deliver learning is discussed in Chapter 4. To maximise the effect of the learning material in Drazah, artifacts should be presented to the player in a salient, easy to understand form. In addition, game features such as feedback and achievements should be used to reinforce positive behaviours and provide the player with an accurate reflection of their ability.

6.3 Educational theories

The work covered in Chapter 4 (*'Evidence for the efficacy and effectiveness of educational games'*) explores the use of various learning theories in existing educational games and discusses how they are implemented

and what effect they have. The most frequently used theory was experiential learning, a possible reason for this is its ability to utilise game features such as exploration and feedback as learning mechanisms.

Section 4.3 looks at behavioural learning, and how best to initiate a change in thinking by gaining a learners attention, and keeping them engaged. The work on engagement and motivation in games (Section 3.4) indicates how best this can be achieved through play.

In this Section, the framework for Drazah's educational approach is developed as a series of mappings from concepts in educational theory to game features. Individually presenting and discussing these theories emphasises which specific educational aspect is being considered and how best this can be incorporated in to Drazah.

In order to effectively present the different aspects of Drazah's learning model, a table containing the concept name and details is used. This presentation technique allows the model to be discussed segmentally throughout the Section. Because there are aspects linking to different learning theories, it is more effective to discuss the individual details of each, before considering the impact of the model as a whole. An example of the table structure is presented below.

Drazah	
Concept	Details

6.3.1 Elaboration Likelihood

The Elaboration Likelihood model proposed by Baranowski *et al* (2008) presents a model of behaviour change through game play in accordance with Social Cognitive Theory (Chapter 4). Baranowski states that gaining a players attention is the first step to initiating learning and behaviour

change, and that emphasising characteristics of Elaboration Likelihood, such as attention and retention, is beneficial in educational game design.

In order to gain and retain a players attention, the findings of Chapter 3 on the effects games have on players is utilised. In particular, the work of Nakamura & Csikszentmihalyi (1990) on motivation and flow in games highlights the importance of challenge in a game, and how this can be used to engage the player. Additional features such as social capital and feedback are used to motivate the player, and present them with information regarding their ability, all designed to keep the player's attention. In Drazah, the attention of the player is gained through ease of play, and the aesthetic components of the client design, discussed in Section 7.8.

The educational benefits of hightened attention, such as increased cognitive functionality (discussed in Section 4.2), reaffirm the importance of the elaboration likelihood model in the development of Drazah.

Drazah	
Elaboration likelihood -	Ensure the game captures the attention of the intended player.

6.3.2 Flow

The principles of Attention, Engagement and Motivation are the initial points of design for Drazah, focusing on gaining the attention of players in order to engage them in play and motivate them through challenge and flow.

The work of Nakamura *et al.* (1990) focuses on the appeal of challenge in activities and the physiological and psychological effects of this on participants. Although not limited to players of games, Nakamura *et al.* states that a common trait of all popular modern-day games is that they produce flow in some way. By presenting challenges to the player that is sufficient

to their ability, videogames are able to enduce a state of hightened attention and concentration in players (Section 3.1).

In an attempt to enduce the state of flow, Drazah must be designed to compensate for the various ability of players, and present challenge that is comparable to the player's skill level. The use of competitive play and achievements are designed to present the player with challenges, in an attempt to facilitate this psychological state.

The importance of establishing flow has both educational and engagement benefits, as players are more likely to acquire knowledge during this state of heightened cognitive functionality (Chapter 4), and enjoy the game and thus become more engaged (Chapter 3). To encourage similar responses to Drazah, an appropriate level of challenge must be catered for, as well as utilising game features that research suggests have motivational and engaging benefits (Chapter 3).

By identifying the importance of attention, engagement and motivation early on in the design phase, it is easier to create specific game features that have shown to encourage these responses (Chapter 3). Further details on the development of motivational and engaging features of Drazah can be found in Sections 6.3.4, 6.5 and 7.8.

Drazah	Flow
Elaboration Likelihood -	Ensure the game captures the attention of the intended player.
Engagement -	Produces the state of flow and can be achieved by presenting challenge.
Motivation -	Engagement and flow encourage intrinsic motivation.

6.3.3 Pedagogy

Educational games can use a variety of pedagogical approaches, one of the more common approaches being experiential learning, or discovery learning (Kebritchi & Hirumi 2008). A pedagogy is an approach or framework for the education of young children, and can be used to present different ways in which to facilitate learning. The advantage of utilising an existing educational framework, such as experiential or instructional learning, is that the process has already been developed and evaluated. Because they have been successfully evaluated as learning techniques, effective implementation of these frameworks increase the likelihood of knowledge acquisition.

The work discussed in Chapter 4 shows that the use of these approaches, with regard to game-based learning is unknown, however the emphasis of these pedagogical techniques (as seen in Chapter 4) in conjunction with the development of an educational game could be significant. Because of this, the design of Drazah's game mechanics is based on the following stages of experiential learning.

Kiili (2004) states that *'Experiential learning consists of the following sequence of events'*

1. Concrete experience
2. Reflective observation
3. Abstract conceptualisation
4. Active experimentation

Experience With regards to experiental learning, 'experience' refers to the process of presenting concepts to the learner and allowing them to

build their conceptual knowledge through interaction (Kiili 2004). By doing so, the learner is able to create their own theories about the artifact, and determine how best to verify their assumptions.

Killi's work on experiential learning in games states that simulations and 'open world' style games provide an excellent platform for allowing the player to gain experience with educational concepts. However, any game can utilise experiential learning by providing the player with the opportunity to experience something new, allowing for a period of reflective thinking and conceptualising before allowing them to experiment.

The variety of play styles and features in modern-day games presents a plethora of experience delivery techniques, by utilising the work of Killi on transparent artifact delivery, Drazah is created to present educational concepts as playable cards. By presenting content in abstract form, Drazah is able to redefine the interaction conventions. Because hazards are presented as cards, they take on the traits of playable cards in a game, rather than a real world entity that should be avoided.

Drazah	**Pedagogy**
Elaboration Likelihood -	Ensure the game captures the attention of the intended player
Engagement -	Produces the state of flow and can be achieved by presenting challenge.
Motivation -	Engagement and flow enforce intrinsic motivation.
Exposure -	New concepts must be presented in an easy to understand manner.

Reflection Once the player has gained a new experience they must be provided with the opportunity to reflect on its attributes and create a mental model. Doing so will help to increase their knowledge whilst preventing any cognitive strain from continual immersion of new information (Kiili 2004).

One of the ways in which this can be done is through reflection, presented as a short period of inaction during play, or through a feedback mechanism.

The importance of this reflection should not be underestimated, as stated by Squire (2003) *'The amount of time spent on reflection should equal the amount of time spent on engaging'*. Games can provide reflection in numerous ways, but it is important that it appears as an in-game goal, rather than a separate mechanism (Paras & Bizzocchi 2005).

The work of (Kiili 2004) and (Paras & Bizzocchi 2005) highlights the significance of reflection in learning in general, and experiential learning specifically. Once the learner is presented with a new concept, they must be provided with an opportunity to reflect upon it. Feedback of actions and performance are ways in which a game can provide information to the player to reflect upon, examples of how this feedback can be delivered in games is discussed in Section 3.2.

In Drazah, the alternating style of play is used to provide the player with time to make a decision about which card they wish to play, as well as feedback on their performance. The use of a round derief stage is specifically intended to provide the learner with an opportunity to reflect on new information.

Drazah	Pedagogy
Elaboration Likelihood -	Ensure the game captures the attention of the intended player
Engagement -	Produces the state of flow and can be achieved by presenting challenge.
Motivation -	Engagement and flow enforce intrinsic motivation.
Exposure -	New concepts must be presented in an easy to understand manner.
Reflection -	Players must be provided with time to reflect on new information.

Conceptualisation It is believed that once someone has gained conceptual knowledge of an artifact, they have understood it. *'The endpoint of acquisition for concepts is when factual or principle knowledge can be used to recognise, identify, evaluate, judge, create, invent, compare and choose'* (Star 2000). Once a player has had sufficient time to observe and reflect upon the characteristics of a new experience they must then conceptualise it. This can be done abstractly (i.e. with regards to its implementation in the game environment) and is the penultimate step in the experiential learning theory. Once a player has gained this conceptual understanding within a game, they are able to create a hypothesis regarding how best to interact/utilise/destroy/solve it and then go on to explore the results of this hypothesis through experimentation.

An example of this appriach n games can be seen in any game that allows the player multiple attempts to complete a challenge. The emphasis of this approach is based on providing the player with the ability to learn through failure. This is done by allowing the player to reflect on the characteristics of the challenge, hypothesise a solution and try again.

Drazah	Pedagogy
Elaboration Likelihood -	Ensure the game captures the attention of the intended player
Engagement -	Produces the state of flow and can be achieved by presenting challenge.
Motivation -	Engagement and flow enforce intrinsic motivation.
Exposure	- New concepts must be presented in an easy to understand manner.
Reflection	- Players must be provided with time to reflect on new information.
Conceptualisation	- Present an opportunity for players to think how best to utilise new information within the game.

Experimentation *'Overcoming problems in a game requires the player to asses their challenge, create a hypothesis on how to beat it, and try again'* (Kiili 2004).

The final stage of experiential learning theory is 'Active experimentation' and can be achieved in games by allowing the player to overcome a challenge through implementing the knowledge they have recently acquired.

A game designed to facilitate experiential learning must be able to provide the player with new experiences, an opportunity to reflect upon them, create a hypothesis and then test these new assumptions. A way in which this is possible is by providing various challenges in which the player must learn new tactics to overcome an opponent. Similarly, repetition of certain concepts can be presented to the player in a manner in which they are motivated to find the best possible outcome through experimentation.

A learning framework for Drazah is founded on the elaboration likelihood model and experiential learning theory, this is done increase the likelihood of knowledge acquisition through a proven method of learning. The work

of Kebritchi & Hirumi (2008) on educational games shows an increased use of experiential learning theory and the work of Squire (2003), Star (2000) and Kiili (2004) help to shape the design of Drazah's learning foundations by emphasising the importance of engagement, motivation and a pedagogy.

Drazah	**Pedagogy**
Elaboration Likelihood -	Ensure the game captures the attention of the intended player
Engagement -	Produces the state of flow and can be achieved by presenting challenge.
Motivation -	Engagement and flow enforce intrinsic motivation.
Exposure	New concepts must be presented in an easy to understand manner.
Reflection	Players must be provided with time to reflect on new information.
Conceptualisation	Present an opportunity for players to think how best to utilise new information within the game.
Active Experimentation	Allow players to use new knowledge to overcome a challenge.

6.3.4 Applying Theory

Elaboration Likelihood and experiential learning theories were applied to the development of Drazah in order to increase the likelihood of learning and to act as a framework for how best to implement educational processes. This is done to shape the games design to facilitate learning and highlight potential design issues specific to educational games.

By considering these educational theories in the initial design stages of the game, it is intended that game features can be tailored towards ef-

fective learning. For example, the Elaboration Likelihood model highlights the importance of challenge, engagement and motivation in educational games.

The work of (Atkins et al. 2002) states the advantages of feedback in games, and how being continually aware of their performance helps players feel more engaged. Common ways in which games gain players attention is through scalable challenge and extrinsic motivation such as social capital (Chapter 3). In order to take advantage of this, Drazah is developed to have scalable challenge in two ways. Firstly, a competitive style of play means that players will often face different opponents, this means that the skill level of their opposition, and therefore level of challenge will vary. Secondly, players are provided with feedback on their performance during rounds, by doing so they are able to gauge their own skill level, and hypothesise on how to do better, thus challenging themselves.

The increased attention required to overcome challenge has additional benefits for engagement and motivation, as highlighted by the work of Nakamura & Csikszentmihalyi (1990) on flow. In Drazah, scalable challenge is produced to help facilitate flow, therefore increasing cognitive functionality and the ability to assimilate new experiences when presented. Although the state of flow is thought to increase intrinsic motivation, the work covered in Section 3.4 highlights ways in which extrinsic motivation is generated in players of games. Features such as player statistics and achievements are created to enhance the players motivation to continue playing. By storing and displaying player information such as name, wins and losses, a social hierachy is created through the desire of appearing compitent (Section 3.5). In addition, the desire to gain achievements, coupled with the motivational advantages of rewarding players positively, inspired the creation of the 10 in-game achievements found in Drazah.

The inclusion of a proven educational theory, in this instance experiential

learning, was applied at the beginning of the development to help shape the game towards facilitating learning where possible. The work of Kiili (2004) on experiential learning in games presents the four stages of this method of learning, and serves as a framework for design. Exposure to new concepts is essential, and should be done with minimal expense to cognitive functionality, requiring games to present new concepts in a salient, natural way.

In order to provide a platform for new concepts, Drazah is made up of multiple situation, hazard and action cards, each designed to present information in an abstract form. An effective card game requires numerous cards, therefore a total of 8 situation, 27 hazard and 27 action cards were produced. By representing these concepts as cards, it is possible to provide information in a salient way (as written information on the card). Because the format of a card game often requires cards to be randomly dealt to players, and removed from play, a surplus of cards is required to ensure sufficient quantity for each game. This increases the total amount of learning material included in Drazah. Each round, a player is dealt eight cards at random (four hazard, four action), this exposure to concepts is informed from the initial stage of experiential learning, *'Exposure'*.

The work of Atkins et al. (2002) highlights the importance of feedback in games, and provides evidence to suggest that this also helps to reinforce the reflection required for experiential learning. Because Drazah is a card game, a natural reflection mechanism would be to assign values to cards, and use these to create a gameplay mechanic such as highest score wins. The inclusion of scores provides the opportunity to present feedback to the player on their performance by displaying their scores each round.

Drazah was created as a card game to effectively repeat the presentation of concepts, provide exposure and increase the amount of artifact combinations. By displaying new concepts as cards, abstract representation

is used, allowing for the game to determine the interaction process rather than relying on real world conventions, meaning hazards can be played as cards, rather than avoided/prevented such as in real life. Additionally, because Drazah relies on exposure and reflection to educate, a card game format provides numerous different combinations of situation, hazard and action cards. By doing so, it increases the conceptual knowledge available through active experimentation, and encourages further exposure through the pursuit of mastery.

6.4 Game mechanics

One of the requirements for the development of Drazah is that it be designed using a sound educational approach, already proven and established through research. As well as providing scope and guidance to the game, the inclusion of experiential learning helped determine the base game mechanics of Drazah, as it was fundamental that the game support and facilitate learning. Repetition and reflection, as well as exposure to new concepts all rely on the games ability to present aspects of learning in engaging ways.

An important facet of card games in particular, and many games in general is scores and scoring. The ability to quantify one's ability and directly compare against an opponent in order to determine a winner is a defining characteristic of a game. The existence of this player comparison relies on a quantifiable way in which to measure ability, often achieved through the application of a scoring mechanism.

Games provide many different metrics in which to generate scores, such as wins and losses, time and experience. In Drazah, a scoring system is implemented to provide feedback on a players ability to correctly re-

spond to a hazard or situation based on combined suitability and severity values. By quantifying the combined suitability and severity of each card combination in Drazah, the game is able to provide the player with relevant feedback on their performance. Additionally, scores can be accumulated in order to determine which player is victorious at the end of the game.

By providing this feedback, it is intended that players will become motivated to beat their opponent by achieving a higher score. In order to do this, the player must learn which combinations of cards will gain them the highest score. The importance, then, of the scoring mechanism is clear. The development of this mechanism is discussed in detail in Section 6.5.

It is intended that the motivation generated by the desire to win will increase engagement and the likelihood of flow, thereby increasing the chances of knowledge acquisition. The work of Kebritchi & Hirumi (2008) highlights the educational advantages of repeatedly exposing learners to concepts, and allowing them to develop conceptual knowledge. Although Drazah's play style already creates repetition, greater duration of play will increase the amount of exposure to concepts and thus provide educational benefits. The work discussed in Section 3.4 helps define the factors that attribute to intrinsic/extrinsic motivation, and looks at how this can be incorporated in games to help increase engagement during play.

One of the key motivational techniques used in Drazah is the use of achievements as rewards, which are used to either reinforce positive actions or encourage longer duration of play via extrinsic motivation. The popularity of achievments can be seen in their inclusion in almost all modern-day games (ESA 2011), and the work of (Ryan, Rigby & Przybylski 2006) demonstrates the appeal of achievements to different styles of players, and how extrinsic motivation through social capital works in games. By rewarding certain behaviours, achievements can be used to reinforce actions through motivation, be it either intrinsically or extrinsically (Section 3.4). In

addition, by providing social capital as extrinsic motivation, it is possible for achievements to generate a willingness to play among players.

The research on social capital, and its benefits on education are an important feature used in the development of Drazah. Achievements and player statistics are designed to be used as social capital, and an incentive for replayability and engagement. By allowing players to see information regarding their opponent, Drazah encourages players to become proficient at the game in order to gain social acceptance with other players through competency. The educational benefits of this are that players spend more time playing the game, being exposed to more of the information contained within each concept and mastering the combinations that result in the highest scores. By doing so they are also learning which actions are most suitable for hazards they may encounter in real life.

6.5 Development of stochastic scoring

The inclusion of statistics in games is not a novel idea, as shown by the work of Shapley (1953) in Section 2.4, and the effective use of real-world data seen in 'Fifa 11' by EA sports. As well as providing a mechanism for the inclusion of data, games can also utilise stochastic design as a scoring system. As stated in Section 6.4, a defining characteristic of a game is the ability to determine which player is a winner (and subsequently another a loser). With this in mind, a stochastic based scoring system is created for Drazah, based on real-world injury statistics, presented to the players as a scoring mechanism.

The Home and Leisure Accident Surveillance Survey (Hass/Lass) is the publication of data regarding the amount of accidents that occurred within home and leisure environments in 2002 (RoSPA 2002). By analysing the

number of people admitted to A&E due to injury, the statistically most dangerous areas become apparent. The use of this data in particular, ensures the situations presented in Drazah are relevant to those that occur in real life.

In order to better simulate realistic interactions of actions, hazards and situations, we requested help from CSEC to help identify the most likely and severe hazards and the most suitable actions to each hazard. An advantage of doing so is that it provides the data used in Drazah's scoring system with real world influence.

In order to calculate the exact values used for each card combination, a list of proposed situations was sent to CSEC for approval. In response, a list of potential hazards and suitable actions were calculated by CSEC for use in Drazah.

The use of injury data is intended to serve as the educational content in Drazah, in addition, the implementation of this stochastic design through a scoring system ensures a player's ability can be quantified. By assigning values to situations, hazards and actions, combination scores can be presented to the player accurately representing the quality of their decision.

6.5.1 Hazard Cards

In order to calculate the score values for each hazard, values between 0 and 10 are assigned to the likelihood of a hazard occurring in a situation (Table 6.5.1), and the severity of the hazard itself (Table 6.5.1). Developed in collaboration with CSEC, values for each hazard and situation were then combined to provide a realistic score.

Once these values are assigned, an overall score can be calculated by multiplying the likelihood and severity values for each combination (Table

6.5.1). In order to provide a more manageable figure for use in Drazah's scoring system, these values are then divided by 10.

6.5.2 Action Cards

To provide real-world benefits to the educational content of Drazah, the use of injury statistics was used. This was mainly to ensure the accuracy of the information but also serves as a way of measuring how successfully games are able to contain and facilitate data.

When dealing with the creation of an educational game, it is important to ensure the accuracy of the learning material, a way of doing this (and the one taken for Drazah) is to collaborate with subject experts. For example, a game designed to teach children about hazards (such as Drazah) should seek guidance where possible to ensure the correctness of any suggested actions to overcome them. This was the primary role carried out by CSEC in this project.

In addition to assisting with the creation of score values representing the likelihood and severity of hazards, CSEC were also able to identify suitable actions for each hazard and produce numerical values related to their suitability in conjunction with every hazard card in the game (Table 6.5.2).

By using this information provided by CSEC, actions recommended in Drazah as highly effective (e.g. those that score the highest), are in fact the most suitable actions for that hazardous situation in real life.

Situation	B1	B2	B3	F1	F2	F3	E1	E2	E3	E4	O1	O2	O3	En1	En2	T1	T2	R1	R2	W1	W2	W3	H1	H2	H3	H4
Roadside	0	0	3	3	7	8	1	1	0	0	1	10	10	1	5	1	1	0	0	1	1	3	3	2	1	1
Canal/Lake	0	0	4	8	8	8	1	0	0	1	1	4	3	1	10	3	1	0	0	10	9	4	1	2	1	1
Living Room	0	0	1	0	3	8	4	6	7	2	2	4	1	0	0	3	2	0	0	0	0	1	8	8	8	5
Bathroom	0	0	4	2	3	7	8	5	5	10	4	2	1	1	0	10	10	0	0	3	3	10	2	7	7	8
Garden	0	0	5	2	5	8	7	6	8	8	9	3	4	1	2	6	6	0	0	4	4	5	7	8	7	2
Garage	0	0	3	2	2	8	8	7	10	5	10	7	4	3	1	10	10	0	0	0	0	3	6	7	7	4
Science Lab	8	9	1	2	2	4	8	7	8	7	7	6	4	10	1	10	10	7	7	2	2	8	9	9	8	9
Kitchen	0	0	9	1	3	7	8	7	10	10	10	2	1	1	1	10	10	0	1	0	0	10	10	10	10	10

Table 6.2: Situations and corresponding hazard likelihood values provided by CSEC

Situation	B1	B2	B3	F1	F2	F3	E1	E2	E3	E4	O1	O2	O3	En1	En2	T1	T2	R1	R2	W1	W2	W3	H1	H2	H3	H4
Roadside	10	5	5	10	6	9	10	6	4	9	6	10	10	6	5	8	10	6	10	10	9	10	5	7	5	1
Canal/Lake	10	5	5	10	6	9	10	6	4	9	6	10	9	6	8	8	10	6	10	10	9	10	5	7	5	1
Living Room	10	5	5	10	6	4	10	6	4	9	6	8	5	6	5	8	10	6	10	10	4	10	5	7	7	5
Bathroom	10	5	5	10	6	4	10	8	7	9	6	8	5	6	5	8	10	6	10	10	5	10	5	5	7	8
Garden	10	5	5	10	6	4	10	6	7	9	8	8	8	6	6	8	10	6	10	10	5	10	9	7	6	2
Garage	10	5	5	10	6	4	10	6	8	9	9	8	9	6	6	8	10	6	10	10	5	10	8	9	7	4
Science	Lab 10	5	5	10	6	4	10	6	8	9	8	8	9	9	6	8	10	6	10	10	5	10	9	9	8	9
Kitchen	10	5	5	10	6	5	10	8	8	9	8	8	9	5	5	8	10	6	10	10	5	10	9	9	9	10

Table 6.4: Situations and corresponding hazard severity values provided by CSEC

Situation	B1	B2	B3	F1	F2	F3	E1	E2	E3	E4	O1	O2	O3	En1	En2	T1	T2	R1	R2	W1	W2	W3	H1	H2	H3	H4
Roadside	0	0	15	30	42	72	10	6	0	0	6	100	100	6	25	8	8	0	0	10	10	27	30	10	7	5
Canal/Lake	0	0	20	80	48	72	10	0	0	9	6	40	27	6	80	24	8	0	0	100	90	36	10	10	7	5
Living Room	0	0	5	0	18	32	40	36	28	18	12	32	5	0	0	24	16	0	0	0	0	4	80	40	56	35
Bathroom	0	0	20	20	18	28	80	40	35	90	24	16	5	6	0	80	80	0	0	30	30	50	20	35	35	56
Garden	0	0	25	20	30	32	70	36	56	72	72	24	32	6	12	48	48	0	0	40	40	25	70	72	49	12
Garage	0	0	15	20	12	32	80	42	80	45	90	56	36	18	6	80	80	0	0	0	0	15	60	56	63	28
Science Lab	80	45	5	20	12	16	80	42	64	63	56	48	36	90	6	80	80	70	42	20	20	40	90	81	72	72
Kitchen	0	0	45	10	18	35	80	56	80	90	80	16	9	5	5	80	80	0	6	0	0	50	100	90	90	90

Table 6.6: Situations and corresponding combined hazard values (out of 100)

Action	B1	B2	B3	F1	F2	F3	E1	E2	E3	E4	O1	O2	O3	En1	En2	T1	T2	R1	R2	W1	W2	W3	H1	H2	H3	H4
Containers preventing hazard from affecting surroundings	10	10	9	0	0	0	0	0	0	1	1	0	0	9	0	0	6	10	10	0	0	2	1	5	1	2
Protective containers prevent substances from affecting the environment	10	10	9	0	0	0	0	0	0	1	1	0	0	9	0	0	6	10	10	0	0	2	1	5	1	2
Step Carefully	0	0	0	8	10	8	1	0	0	0	0	2	1	0	2	0	0	0	0	1	9	7	1	1	0	2
Be careful when devices are turned on and do not use near water	0	0	0	0	0	0	1	0	10	10	3	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0
Use biological protective gear and avoid hazard	1	9	4	0	0	0	0	0	0	0	0	0	0	2	0	1	6	0	0	0	0	0	0	0	0	0
Call 999	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	10	0	0	0	10	0	1	9	0	0	0
Stay clear and call 999	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	10	0	0	0	10	0	1	10	0	0	0
Be careful around or near large/heavy objects	0	0	0	0	1	0	0	0	2	0	0	10	8	0	2	0	0	0	0	0	0	0	5	0	0	0
Child resistant bottles and locks	0	2	0	0	0	0	0	0	5	0	5	0	0	0	0	0	8	0	0	0	0	0	0	0	5	0
Use protective clothing and avoid spending time near the hazard	1	9	4	0	0	0	0	0	2	0	1	0	3	0	1	0	1	1	10	0	0	0	0	6	0	0
Wash germs off with hot water and soap	0	2	10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Move or avoid Potential hazardous objects	1	1	6	0	2	10	1	0	1	1	3	8	6	0	3	0	2	1	1	0	1	7	1	5	1	0
Stay away from the hazard	0	1	4	10	5	4	1	2	0	1	1	8	6	2	3	0	1	1	5	1	6	8	5	5	0	8
Use a rope, scarf or branch to try to reach the person	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	1	0	0	0	0	0
Keep Distance	1	1	1	10	5	2	1	2	1	1	1	8	6	2	3	0	1	1	5	1	7	8	5	5	0	8
call 999	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	10	0	0	0	10	0	1	9	0	0	0
Lock up hazardous objects and keep them out of reach	3	6	1	0	0	0	0	0	5	0	5	0	0	0	0	0	10	0	0	0	0	0	0	8	5	0
Call 999	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	10	0	0	0	10	0	1	9	0	0	0
Dispose of waste properly and report litter and pollution to land owner	5	5	4	0	0	1	0	0	0	0	1	1	0	10	10	0	2	0	0	0	1	1	0	5	5	0
Prevent flammable objects from coming in to contact with heat	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	7	10	0
Do not touch dangerous objects	0	2	4	0	0	0	1	5	1	2	1	4	0	3	3	0	1	0	0	0	1	0	1	8	0	8
Keep out of reach of children	0	0	0	10	10	0	1	3	10	0	10	10	8	2	7	0	10	2	6	0	10	8	7	10	10	10
Use dangerous objects carefully, taking appropriate safety precautions	7	7	7	0	0	0	0	0	8	4	10	1	2	0	0	0	1	1	5	0	0	0	0	10	1	10
Keep area clear of mess and clutter	0	0	0	0	5	10	0	0	1	0	1	1	2	0	4	0	0	0	0	0	8	0	0	1	0	1
Wear shoes with good grip and walk carefully	0	0	0	1	5	10	0	0	0	0	0	1	0	0	0	0	0	0	0	0	10	7	0	1	0	0
Watch out for moving objects	0	0	0	1	1	1	0	0	5	0	2	1	10	0	0	0	0	0	0	1	5	2	0	0	0	0
Avoid hazard and shut off power if possible	0	0	0	0	0	0	8	10	0	8	0	1	0	0	0	0	0	0	0	0	0	6	1	0	0	0

Table 6.8: Actions and corresponding values provided by CSEC

6.6 Early evaluation through paper prototyping

The educational benefits of engaging flow and the heightened attention and engagement attributed to being in this state highlight the importance of ensuring its occurrence in educational games. As stated in Section 3.1, it is important that the person experiencing flow is able to concentrate on the challenge they are partaking in. One of the ways in which this concentration could be disrupted is through playability issues, whereby the player is forced out of immersion due to an inability to continue.

Due to the relitively long development cycle of an electronic game, it was considered more efficient to conduct a playability evaluation on a paper version of the game, before developing the software that would become the digital version.

By conducting this evaluation, potential gameplay issues can be discovered and fixed before full implementation. The primary objective of the development and evaluation of a paper prototype was to ensure the game itself was playable and engaging by testing the mechanics through multiple play-throughs.

6.6.1 Prototype results

One of the main issues encountered during the prototype phase was which combinations of cards were available to the player due to the fact that each hazard card has a matching action card, and only those can be played together. This is considered a significant playability issue because Drazah randomly deals cards to players, meaning that it is possible for a player to reach a state where they can not continue the game through no fault of their own. This renders the player unable to carry out an action, and therefore the game becomes unplayable in this state.

A solution to this issue is to expand the potential combinations of cards to allow any hazard cards to be played regardless of their applicability. This freedom of choice however can have a negative impact on a game, as it can remove the player's motivation to pick the best card, and subsequently not need to learn the values of card combinations in Drazah. However, due to the inclusion of the stochastic scoring mechanism, different scores are awarded to each combination, thus reinforcing the players motivation to learn the highest scoring combinations, allowing them to win the game.

By assigning scores to every combination of cards in the game, players must continually learn those that will score the highest in order to beat their opponent. Although a player is free to select and play any card in their deck, it is more beneficial for them to consider which will score them the highest, thus restoring the motivation to think about their actions before carrying them out.

The outcome of the playability evaluation helped to identify a significant issue with the design of the game, in response to this, the game design was altered to allow greater playability and increase the likelihood of inducing flow.

Chapter 7

Design of the software

7.1 Introduction

The main contribution of this project is the development of Drazah, a digital, multi-player educational card-game built on the principles of experiential learning and social cognitive theory. Chapter 6 focuses on the creation of Drazah from an educational standpoint, this Chapter focuses on the development of Drazah as software.

Sections 7.2 and 7.3 highlight the requirements for Drazah and the reasoning behind the selection of the platform used for development.

Drazah's system architecture is then presented in Section 7.4 followed by a definition of the protocols created for communication between the server and its clients.

Section 7.4 presents the client/server architecture developed for Drazah. The game and its state are controlled by the server while the user interaction and presentation of game state happens on the client. Details of how the server was designed and created, including communication protocols,

server discovery and game concurrency can be found in Sections 7.5, 7.6 and 7.7 respectively. In addition, details of the development of the client Graphical User Interface (GUI) are presented in Section 7.8.

7.2 Requirements

Drazah is an alternating turn-based game, where players take turns in selecting cards to progress play to a final end state. Each game consists of 8 rounds, each consisting of a Situation, Hazard and Action card. As described in Chapter 6, Drazah was designed to be a two-player, turn-based card game that allows players to utilise characteristics of hazards and actions to overcome their opponent. As well as providing a method of delivering gameplay and user interaction, the software must be designed to support Drazah's game mechanics.

The work of Atkins *et al.* (2002) highlights the performance benefits of using salient, digital feedback mechanisms, such as those seen in modern-day video games (Section 3.2). The main advantage being that because feedback is provided in an electronic form, it is easier and quicker to present players with an accurate assessment of their performance via game feedback such as player health, score or responses to correct answers. In order to maximise the amount of feedback presented to the player (for both engagement and educational purposes (Section 3.2) a 'review' stage was created at the end of each round. The advantages of having a review stage during play is that it provides an opportunity for feedback, and to enable the player to reflect upon their performance. The importance of this is considered in the development of Drazah, and was also a contributing factor in deciding to make a digital version of Drazah.

One of the significant advantages of a digital version of Drazah is the po-

tential to increase the amount of game exposure by integrating in to a Virtual Learning Environments (VLE). To help achieve this, the game is developed in a cross-platform language so that it can run on different operating systems if required. Additionally, the system architecture and communication protocols must be adaptable enough to work on a variety of devices, and easy to use since the game is designed to be played by children.

Further information on the system architecture used in Drazah can be found in Section 7.5, details on the development of Drazah's graphical game elements are presented in 7.8.

7.3 Platform

In order to meet the requirements of Drazah, the software must be developed to utilise a client/server architecture; must be cross-platform to increase potential players; and be easy to develop and implement. To help achieve this, Drazah was developed using 'Python', an open-source, cross platform programming language (Foundation 1990).

Python is one of many 'High-level' programming languages, enabling users to create applications using abstract representation. In Python, examples of these representations include functions, variables and objects. The use of these features are intended to make programs easier to debug and faster to develop with.

Python is an object orientated programming language designed to be easy to debug, portable and cross platform (Windows/Linux/Mac OS X). The advantage of Python over other high-level languages, such as 'C', is that it is object orientated, and allows for the expression of ideas more succinctly. Because of these features, and the preference of the author, it was chosen as the language for the development of Drazah.

In order to utilise audio and visual outputs, many high-level languages use the 'Simple DirectMedia Layer'(SDL). The SDL is an open-source, cross-platform media library designed as an interface for sound and graphics devices. Although the SDL is extensive, due to the visual nature of modern-day game interfaces, including the one created for Drazah, additional functionality is required to enable the user to effectively interact with on-screen graphics.

Pygame Pygame is a collection of Python modules specifically designed for the development of games. Pygame adds functionality to the SDL that allows the user to create games and multimedia programs in the Python programming language (pyGame 2000).

The Pygame library is able to load and display images of varying size and format, and allow for the user to interact with objects by registering mouse position and clicks. These features can be used to develop a GUI for Drazah by displaying various game items such as cards and scores on screen. Because of this, Pygame is used for the development of the client.

SQLAlchemy As well as a server and client, Drazah is also required to keep records of a player such as overall wins, losses and achievements. During play, the server is required to retrieve a player's information, update it and save it. These actions are carried out using Structured Query Language(SQL), however, because the game server is written in Python, there needs to be a method for interpreting these actions in the Python programming language. This is the primary function of the SQLAlchemy Python library in Drazah.

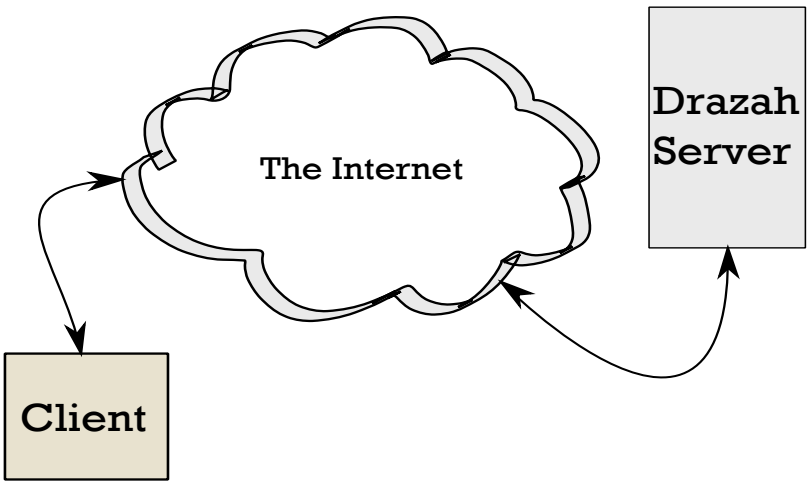

Figure 7.1: Diagram representing system architecture

7.4 Client/Server architecture

A versatile deployment strategy will allow schools to host their own game of Drazah via a Virual Learning Environment (VLE) and still allow CSEC and RoSPA to host their own publicly accessible game server.

A standard network connection will allow clients to connect to the Drazah server remotely, and is therefore the most suited method of communication between the client and server (Fig. 7.4). The advantages of this is that the Drazah server address can be changed to operate on either a local or public server. This can be achieved with very little change to the games parameters, and would allow schools to host their own games and CSEC & RoSPA to host public ones.

To allow clients to easily connect to the nearest game server, a controller class was designed to broadcast the server address across a network via

Universal Plug and Play(UPnP) and accept client connections (Section 7.6.2). Once a player is connected they are placed in a waiting list, the controller then creates a game object, taking two players from this list and connecting them to that game (Figure 7.5).

7.5 Server Design

The choice to use an alternating, progressive style of play for Drazah meant that a state machine is ideally suited to implement the logical steps of the game set as 'states'. Input from the player is required to transition the game between states, once this is done, the client receives new information via a network communication.

A finite state machine is a behavioural model used in software design to build logical state-progression devices, based on a finite number of states, linked through actions known as transitions. Generally a change in state is triggered by an event or condition and will cause the system to either progress to the next state or return to the same state, depending on the pre-programmed behaviour (Martin 1998).

The use of a state machine can help in the development of complex systems because they require the designer to consider all possible states and transitions before development (van Bergen 2011). There are several advantages to this approach. It can uncover potential design issues early on in development, by forcing the designer to consider all possible states and transitions before implementation. Because all aspects of state are represented by the state-machine on the server, the client does not need to independently maintain and update its own state, simplifying client development and making it impossible for clients and the server to move out of sync.

FSM's are an effective way of creating systems that rely on a natural progression of a finite number of actions. The state machine designed for Drazah specifically deals with the current state of a game that is running, preventing two connected clients from being in different states of the same game. An example of how a game transitions through states is provided in the following paragraph.

A game object within Drazah will remain in its default starting state is **NEW** until *2 player connections are received*, this is the state's only action. Once these connections are initialised, the server will enter the state **GETINFO** and, using each players name, will *retrieve the corresponding records from the player database* (if there aren't any, then a new record will be created). The game then moves through the **W_Pn_S**(*Waiting for server to select Situation card*), **W_Pn_H**(*Waiting for player 'n' to select Hazard card*) and **W_Pn_A**(*Waiting for player 'n' to select Action card*) states, transitioning when a player selects the appropriate card for that state (*Select Situation Card, Hazard Card, Action Card*). Once the round debrief stage **RDB** is initiated, the game will either start play again by transitioning to state **W_Pn_S** or end the game by entering the **END** state this transition is entirely dependant on the amount of remaining situation cards (*Situation Deck > 0, Situation Deck = 0*). Once in the **END** state, the game will tell the client to present the 'play again' and 'quit' options to the player. Regardless of the response from the first client, the game will stay in the **POSTGAME** state, if the first player selects 'quit' then the connection to that client will be closed, if they select play again their connection will be recycled in to the player waiting list. The response to the second client will be the same, however after both client connections have been closed or recycled, the game will enter the **DELETE** state, where it is closed by the controller.

The primary function of the RDB state is to provide feedback to the player regarding their performance. On completion of a round (one of eight a

State	Description
NEW	Server has received two player connections and starts a new game.
GETINFO	Retrieve each player name, and check Database for previous records.
W_Pn_S	Server deals cards, and selects the first Situation card to be played (server always plays situation card)
W_Pn_H	Instructs one client to highlight Hazard deck so player can make selection.
W_Pn_A	Instructs one client to highlight Action deck so player can make selection.
RDB	Round has finished, display the cards played and their combination scores, wait for eight seconds, then update state to END.
END	Check value of Situation Deck, if there is any left, then start a new round, otherwise move to POSTGAME state.
POSTGAME	Wait for client responses or time out, any players that select 'play again' get recycled to the controller..
DELETE	End state, the controller will remove any game instances in this state.

Table 7.1: Drazah States & Descriptions

game) the server will enter this state and instruct the client to display the values of each card combination. Once the server has waited the determined delay time (to allow each player to reflect upon their actions) it will update its state to start a new hand.

When transitioning to a new state, the server will inform the client via a game update communication containing information such as state and score. Further details of this update can be found in Section 7.6.1.

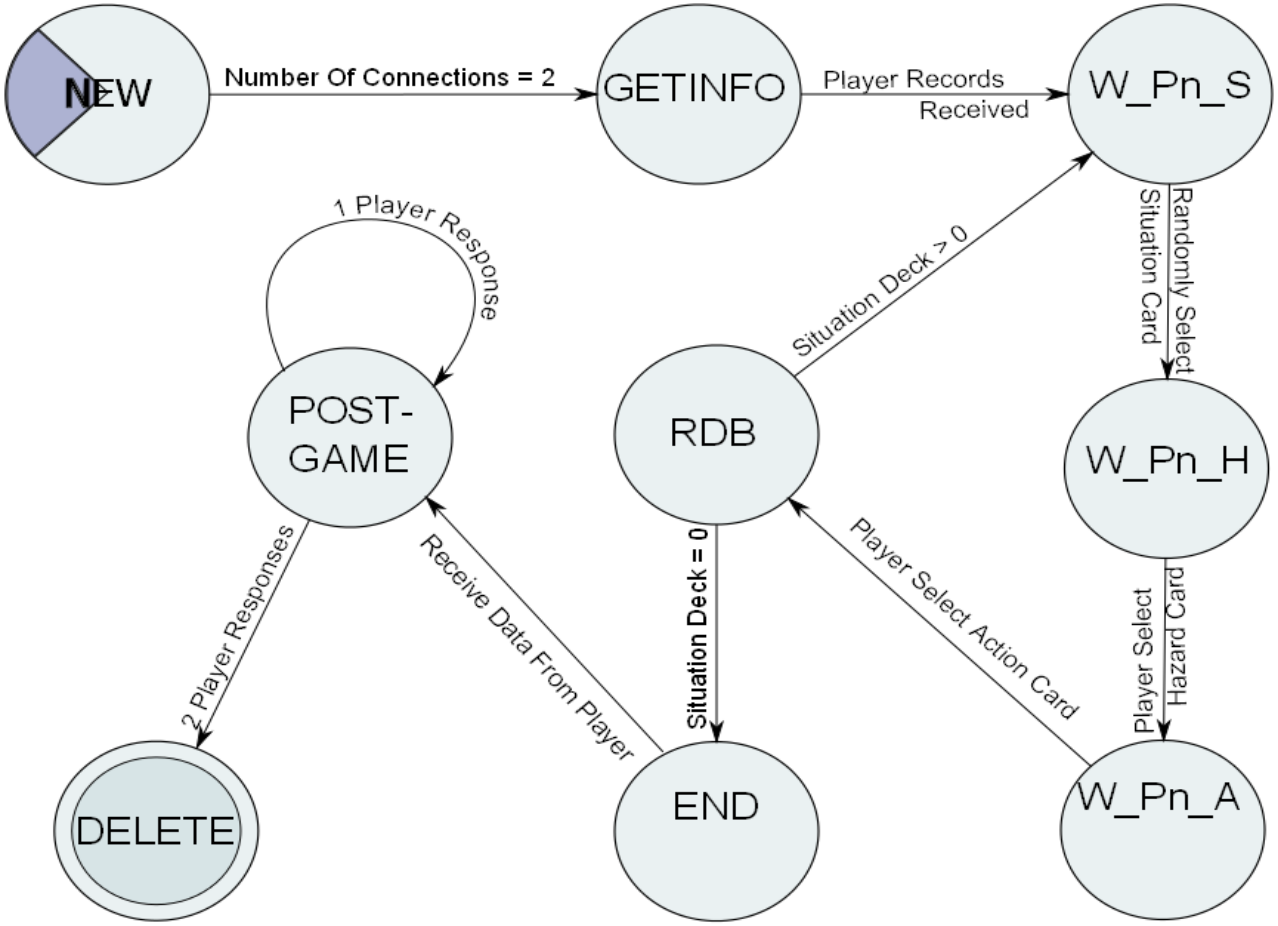

Figure 7.2: Drazah State Diagram

7.6 Communication protocols

Communication between the server and clients is essential to be able to provide status updates and game play. The way in which Drazah sends communications is via a TCP/IP network connection, using a destination's 'address' (Such as domain name or IPv4 address) and port number. Although it is possible to host the server on a publically accessible server, for the purpose of development, Drazah was hosted and tested via local network connections.

Because Drazah is designed to run on either local or public networks, players are able to connect using their local network connection or the Internet. To ensure connecting to a local server was easy, an address broadcast program was created for automatic server discovery and connection. This program broadcasts the server address and port number across the Local Area Network (LAN) as a data packet using UPnP. (Section 7.6.2), an example of the syntax of this communication is presented in EBNF in Figure 7.4.

7.6.1 Protocol Syntax

The Drazah server communicates to clients by sending packets of data containing a single string, this is then seperated in to individual tokens by the client, according to their prefix, and used to update the state of play. Depending on what state the server is currently in, different actions will be available to each client. This state is communicated to the clients as one of the pieces of information in the data packet.

The server sends multiple pieces of information within a single string, this includes information on the following criteria:

- Which cards are in the players Hazard Deck
- Which cards are in the players Action Deck
- The current 'Stack' (Which Situation/Hazard/Action cards are currently in play)
- Current score
- Check to see if any new achievements have been gained
- Total wins
- Total losses
- Total draws
- Opponent details (wins/achievements/score/etc)

Each time the game server enters a new state, it will send all the required information to each client, providing them with everything needed to transition the server to the next state. Because the server controls the games

$$
\begin{aligned}
\langle \text{Status} \rangle \quad \models \quad & \langle \text{HDeck} \rangle \langle \text{newline} \rangle \\
& \langle \text{ADeck} \rangle \langle \text{newline} \rangle \\
& \langle \text{Stack} \rangle \langle \text{newline} \rangle \\
& \langle \text{Score} \rangle \langle \text{newline} \rangle \\
& \langle \text{State} \rangle \langle \text{newline} \rangle \\
& \langle \text{Achievements} \rangle \langle \text{newline} \rangle \\
& \langle \text{Wins} \rangle \langle \text{newline} \rangle \\
& \langle \text{Losses} \rangle \langle \text{newline} \rangle \\
& \langle \text{Draws} \rangle \langle \text{newline} \rangle
\end{aligned}
$$

Figure 7.3: *SendStatus* communication

state, the client never needs to control its state, this makes the client considerably easier to create, and makes it practically impossible for clients to be out of sync with each other or the server. Because the server controls what the client displays and what actions the player is able to do, all of the information required must be sent when the server transitions between states.

In order to easily identify the data contained within the update, a prefix is assigned to each section of the string (such as 'STATE:'). The client is then able to identify each prefix, and and update the client display with the relevant information (e.g. 'STATE:NEW').

Extended Backus-Naur Form (EBNF) is a notation for formally describing the syntax of context-free grammars, such as programming languages, and is an extension of the original Backus-Naur Form (Pattis n.d.). EBNF is used in this project to describe the syntax of Drazah's communication protocols, included in the Appendix A.

During a game of Drazah, several different communications are sent to the client, the purpose of which are to update state or player information. During the GETINFO state, the server requests the players name so that it can check the database records, if a match is found, the server sends the client the player's information. The syntax for these communications are shown in Appendix A.3 and A.4. The game will then enter the state W_Pn_S in which a Situation card is randomly selected, this is then sent to the clients, along with a request for one player to select a Hazard card (Section A.6 and A.7). During the card selection states (W_Pn_H, W_Pn_A) the 'AskCard' communication is used to request a card from the player. To allow the client to display the relevant information during the RDB state, the server sends the round stack and scores via the 'Debrief' communication (A.8). Expressions commonly used in Drazah communications, such as *<String>* & *<Int>*, are presented in Appendix A.10.

7.6.2 Server Discovery

Drazah is required to be easy to play, and easy to find opponents and games, this is because it is designed for young children, with varying levels of computing ability. It is therefore beneficial if the client is able to connect directly to a game without having to search for, or manually input, an address. In order to do this, a dynamic server broadcasting program is integrated in to the server controller that uses Universal Plug and Play (UPnP) network protocol.

By concatenating the server's network address (IPv4) and port number to a single string, and using the same method of prefixing data packets with relevant keywords, Drazah is able to broadcast its address across a network using UPnP. This data packet is then discovered by the client, which identifies the prefix and uses the information to automatically connect to the game server (Section 7.4).

This means that any clients activated on a local network that have the automatic server discovery program running will automatically be adding to that servers Player waiting list.

$$
\begin{aligned}
\langle \text{Broadcast} \rangle &\models \text{`}DRAZAH:\text{'} \langle \text{<IP>} \rangle \text{`}:\text{'} \langle \text{<Port>} \rangle \\
\langle \text{<IP>} \rangle &\models \langle \text{<num>} \rangle \text{`}.\text{'} \langle \text{<num>} \rangle \text{`}.\text{'} \langle \text{<num>} \rangle \text{`}.\text{'} \langle \text{<num>} \rangle \\
\langle \text{<num>} \rangle &\models \{< int >\}^3 \\
\langle \text{<Port>} \rangle &\models \langle \text{<string>} \rangle
\end{aligned}
$$

Figure 7.4: Server Details

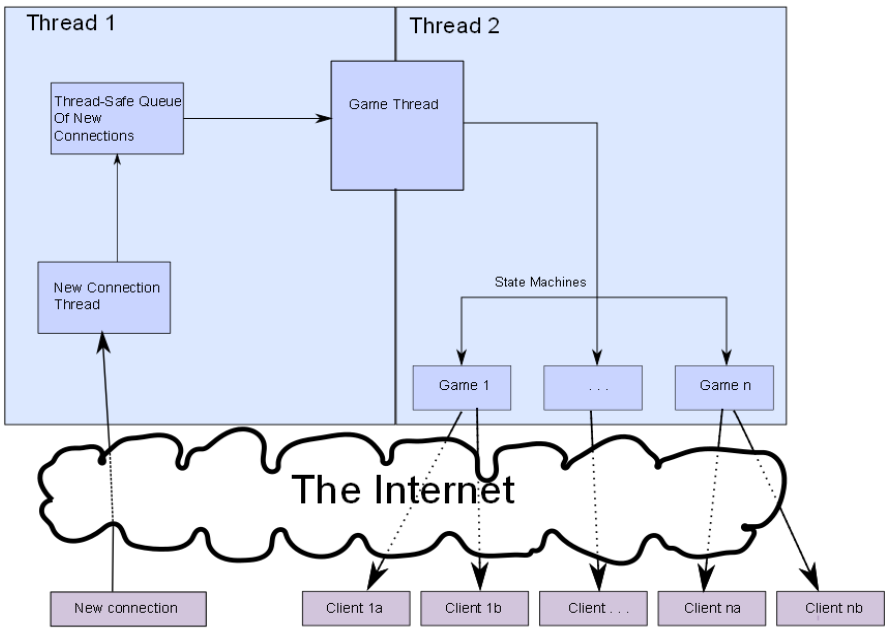

Figure 7.5: Concurrency Diagram

7.7 Game Concurrency

Rather than creating processes or threads for each game, Drazah's server was created to act as a controller for multiple instances of games running through the pre-programmed states of play (Figure 7.5). When a client informs the game instance that an action has been taken (e.g. Hazard Card Played) the game will communicate a state progression to both clients. This means that communication is only required on state transitions, making use of the buffered Transmission Control Protocol (TCP) communication protocol.

The main function of the games state machine is to control what events

progress a games state, and determine what information is sent to clients. When a state change is initiated, the relevant actions can be taken without having to prioritise which game requires action first. For example, if there are two games running, the first (Game 1) is currently waiting for a player move, whilst the second (Game 2) has just finished a hand, the server will prompt a state update to all clients, allowing Game 2 to transition to the state 'round debrief'. Because all game instances are controlled by a single game thread, a state update communication can be sent to all games, allowing clients to transition state once certain criteria is met. Once the server has progressed a games state, the clients are updated to display the relevant information (e.g. deal new cards).

One of the advantages of this method of communication is that there does not need to be a continual transfer of data between the client and server. Additionally, there is no need for clients to communicate with each other directly, significantly reducing the complexity of the network topology and traffic.

7.8 Client Design

Although the mechanics of the game have already been designed (Chapter 6), a client is required for the electronic version in order represent the game and allow user interaction. By utilising features such as text boxes, as well as image and colour display, an graphical game interface can be created for players to interact with.

This section covers the creation of the game client, including the development of the interface through the creation of artifacts such as cards and achievements. Additionally, the protocol for client-to-server communication is explained in Section 7.8.1.

State	Client action/response
NEW	Send name
GETINFO	Display opponent information
W_Pn_S	Display Situation card and prompt player to select Hazard card
W_Pn_H	Display Hazard card and prompt player to select an Action card
W_Pn_A	Display Action card
RDB	Display Scores
END	Play Again or Quit

Table 7.2: Client state responses

7.8.1 Server Communication

The client is designed to periodically update the server with information regarding the games current state via a string formatted data packet, this is done when certain criteria are met (such as a player selects a card). The advantages of this method of communication has been discussed previously.

The way in which this communication works is similar to the prefix identification discussed in the section on protocol syntax 7.6.1. By analysing client responses, the server is able to update the game's state depending on where in the game sequence the client is. When a state update is received from the server, the client executes the relevant function, usually either updating a score or presenting the player with an action to take.

Because the client holds no state of its own, it is used primarily to display the information received from the server, and allow the player to make choices. This minimalistic client design means that different versions of the client could easily be implemented by schools and/or RoSPA/CSEC.

Prefix	Content
⟨Name⟩	⊨	$'NAME:'$ ⟨<pName>⟩
⟨pName⟩	⊨	⟨<string>⟩
⟨Play Card⟩	⊨	$'PLAY:'$ ⟨<Card code>⟩
⟨Card code⟩	⊨	⟨int⟩
⟨End⟩	⊨	$'END:'$ $'PlayAgain'$ | $'Quit'$

Figure 7.6: Client-to-Server communication protocol

7.8.2 Interface

The client interface is entirely determined by the file *Client.py* (Appendix B), this helps to reduce the complexity of the server because there is no need for the server to store data regarding the client interface. Additionally, because the client interface is separate from the server, new clients can be developed for Drazah, providing they follow the communication protocols and game format. As the clients primary function is to display data received from the server, the method of doing this can be changed. The development of a GUI is carried out in this project, however different versions such as text-based or even clients developed by CSEC & RoSPA could be implemented easily. Due to the nature of Drazah's system architecture, different types of clients could be used within the same game instance, therefore increasing the amount of total players.

Because Drazah runs as a state machine, it requires some event for state change to occur. In most cases this will be based on user input, to do this a Graphical User Interface (GUI) is created to enable the player to complete actions and subsequently transition the game through the states of play. In addition to transitioning through play, certain features are required by the

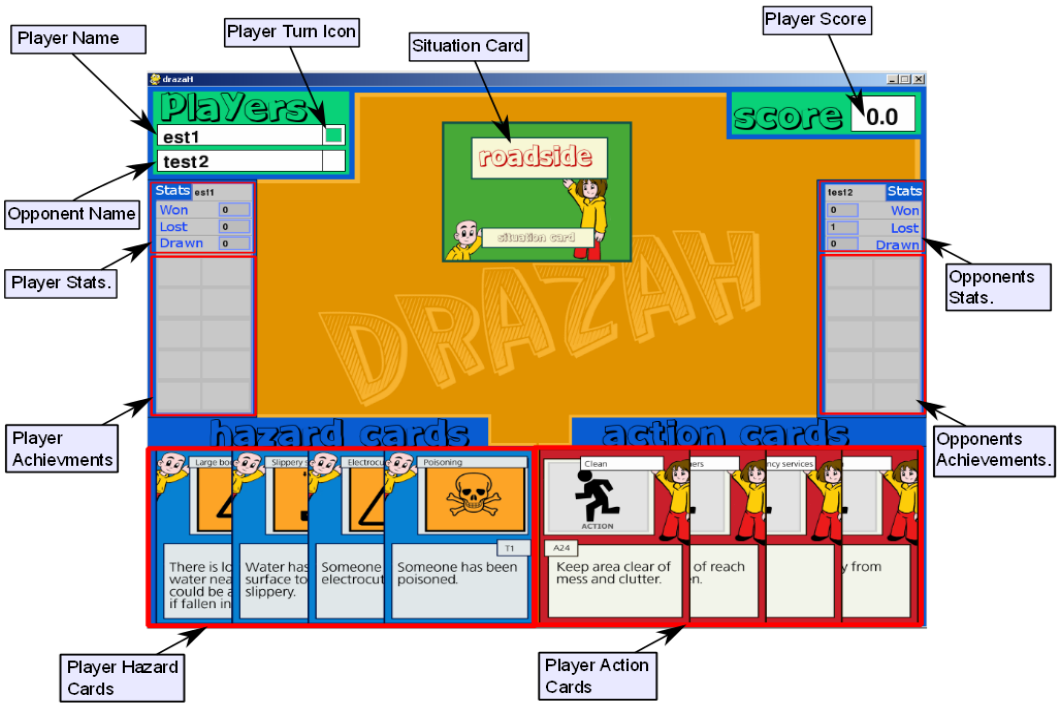

Figure 7.7: Drazah client

client in order to fulfil expectations of modern-day games. One of these features is the ability for the player to allocate themselves a name.

Player Name In order to allow players to enter a name, a text-entry box is used to capture the inputted data, this is presented to the player at the beginning of play in the 'Lobby'(Fig. 7.8). In addition to providing players with a way in which to identify each other, the use of player names as unique identifiers allows them to be used as primary keys so that game can retrieve a certain player's statistics from the database via SQL. This is done by identifying the entered text, and retrieve the relevant data, if the name is not found then a new record is created in the database (Table 7.3).

Field name	Data type
ID	Integer
Name	String (Primary-key)
Wins	Integer
Losses	Integer
Draws	Integer
Score	Float
Achievements	Integer

Table 7.3: Player Database

Lobby Drazah recruits players through a waiting list, therefore it is possible that a player who has passed the 'lobby' screen may have to wait for the server to identify another client connection before it can initiate a game. In an attempt to reduce the likelihood of disengagement during this period, an animation can be used to inform the player that the game is in the process of finding an opponent. In order to do this, pyGame is used to display a series of images, the client displays each image for a short period before moving on to the next, giving the effect of an animation being displayed (Fig. 7.9).

Cards Game artifacts such as cards and achievements are created as independent images and displayed in the client using pyGame, the advantage of using image files is that these images can be altered without having to alter the code used to create the client. Once the game server has randomised the two decks of cards, they must be dealt to the player, the client receives this data packet (Section 7.8.1)and displays the matching image files.

Each deck of cards (*Situation, Hazard or Action*) have a colour-scheme, allowing the player to quickly identify which category the cards belong to. Because Hazard and Action cards have several pieces of information on them that need to be easy to read, they follow the same design patterns

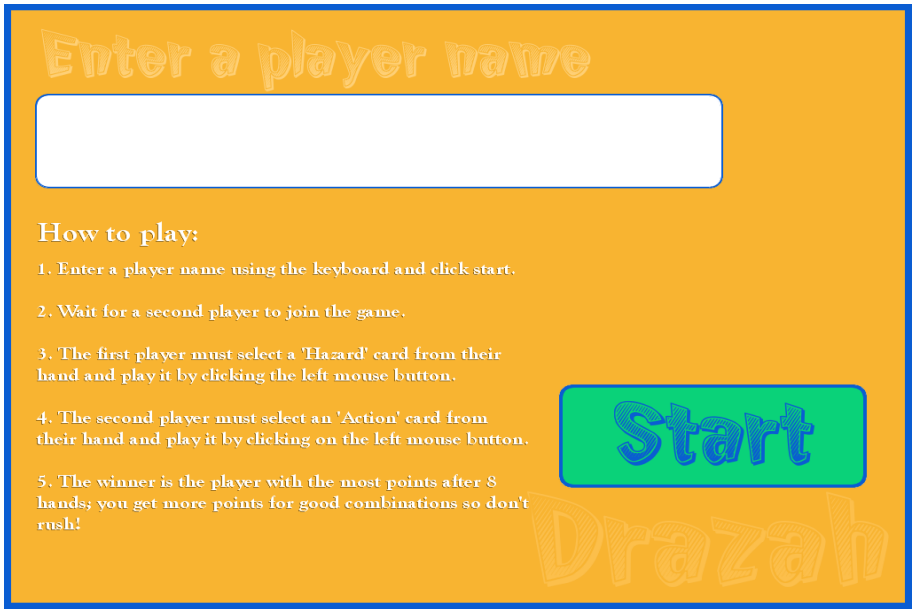

Figure 7.8: Drazah Lobby

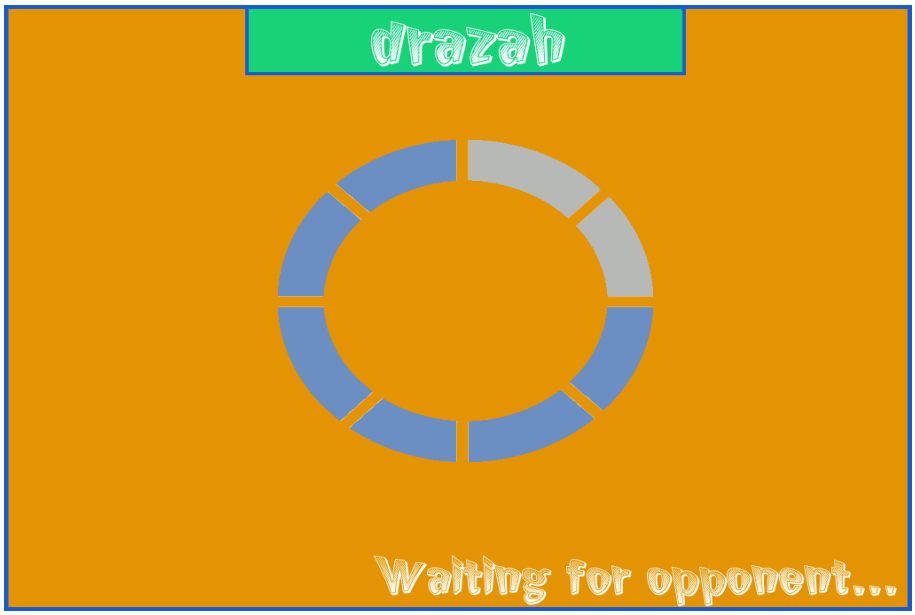

Figure 7.9: Drazah Waiting Animation

and information placement, creating a convention the player gets used to. (Fig. 7.10).

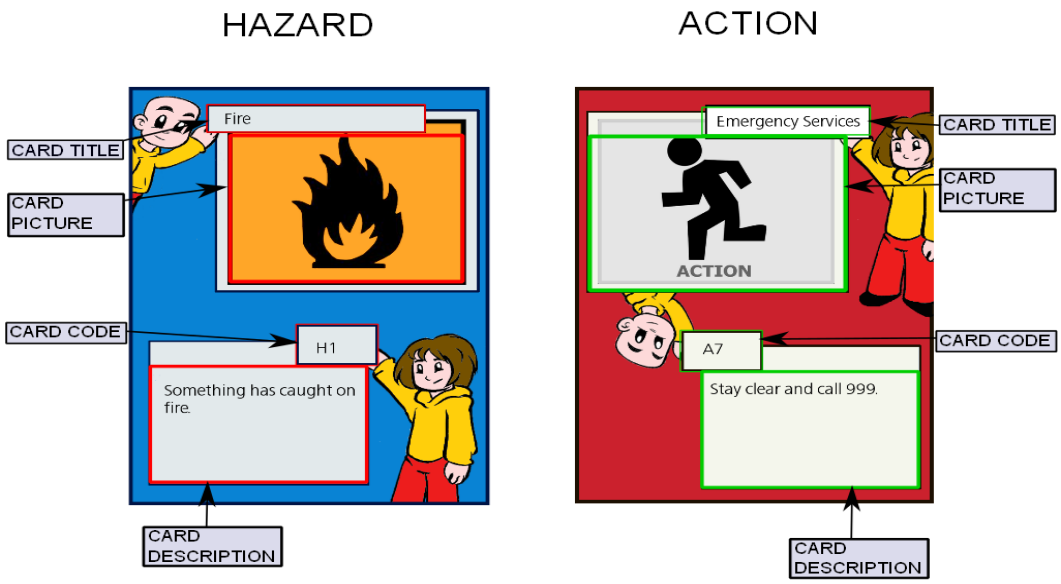

Figure 7.10: Hazard and Action Cards

To help the player identify which type of card they are supposed to play, a ‘highlight’ function has been created to determine which card a player is meant to be playing at a certain stage of the game, and allow them to move upward ten pixels if the mouse cursor moves over them.

Stack To complete a round of Drazah, hazard and action cards must be played by the players, because there is no situation deck, one is selected by the game at the beginning of each round. Once each of these cards are played they are displayed in the ‘stack’ to both clients, meaning players are able to see which combinations are in play and to best decide on the

Figure 7.11: Drazah Stack

next move. In order to do this, each card that is played is removed from the list of available cards, and appended to the stack display.

Round Debrief To meet the requirement for a salient feedback mechanism (Section 7.2), a round debrief stage is included at the end of each round. This provides each player with an opportunity to reflect on the card combinations and scores, in relation to the experiential learning requirements stated in Chapter 6.

Drazah's round debrief state is initiated between each round, and lasts for eight seconds. During this time, players can actively reflect on their performance via their scores. To help visually emphasise these scores,

Figure 7.12: Drazah Round Debrief

the background of the game is partially greyed-out, so that only the current stack of cards and scores are presented in full colour. This is done to draw attention to the scores and make them easier to read. Figure 7.12 shows this game state.

On completion of a game, each player is given the opportunity to play again or quit. If only one player decides to play again, they are recycled into the player waiting list and the waiting animation is initiated again. From here, they will be selected along with another player to begin a new game.

Motivation The competitive aspect of Drazah is intended to produce challenge, and ultimately engage the players in flow. The game provides

variable challenge by selecting players at random from the waiting list. Similarly, the large amount of concepts and combinations of cards means that players are often interacting with new material, and having to continually update their conceptual knowledge by experimenting with new card combinations, it is intended that this also provides challenge and engages flow.

In addition to challenge, achievements and social capital are exploited to increase player motivation in Drazah. The work in Section 3.4 looks at motivation, and Chapter 6 explains how these are built in to the game mechanics.

Achievements Achievements are designed to incentivise certain aspects of play and reinforce extrinsic motivation in players, their popularity can be seen in their inclusion in almost all modern-day videogames (ESA 2011).

In games, achievements are often implemented as collectible rewards obtained when a player fulfils certain criteria. These are often presented in a form that is relevant to the game environment or setting (for example a racing game may present rewards as new cars), alternatively, rewards can be presented as medals or trophies, reflecting the nature of them being awarded for a persons achievement in real life.

Achievements have become an important feature of many new games. For this reason, and their motivational benefits to play and learning (Section 4.2), they should be considered during the design of any educational game.

In Drazah, achievements are awarded in the form of medals, these are then displayed in the player info section of the screen, this is done so that an opponent can see which achievements a player has gained. The motivation behind sharing player statistics comes from the work on social

Hazard Number	Reward Criteria
0	Score 10 playing a Hazard card
1	Score 10 playing an Action card
2	Win a game of Drazah
3	Win 10 games of Drazah
4	Win 50 games of Drazah
5	Achieve a win-to-loss ratio of 2.0 or higher
6	Win a round using a 'Water' class Hazard
7	Win a round using a 'Heat' class Hazard
8	Win a game with an overall score of 30 or higher
9	Win a game with an overall score of 60 or higher

Table 7.4: Achievement award criteria

capital in games (Section 3.5) and the work on Self Determinate theory (Section 3.4).

In Drazah, there are a total of ten achievements for players to collect, each of which require the player to complete certain criteria (such as score ten with a Hazard card). The criteria for each achievement was designed to increase the amount of game content the player would interact with, thereby increasing the coverage of the game.

To record achievements, an achievements field was added to the player database, allowing the game to record what the player had already been awarded. During play, these achievements will then be displayed by the client.

The server checks for new achievements during the play states (W_Pn_H, W_Pn_A) and inform the client to display any new achievements during the round debrief state initially, and then in the player information section afterwards. The achievement number is used to store a players achievements as a bitmask as follows.

Adding an achievement to a player's bitmask:

$$achievements' = achievements \vee 2^n$$

Checking for an achievement in a player's bitmask:

$$hasAchievement(n) = achievements \wedge 2^n \neq 0$$

Chapter 8

Conclusion

In this thesis, a range of pedagogic styles, theories and applications have been examined. In each case, the applicability of the theory to game-based learning has been considered . Additional research was conducted on the efficacy of using games as a digital learning platform, the effects of games on players and how best to deliver learning material through games (Chapters 3 and 4).

Much of the related research suggests that using games in educational is feasible, useful and desirable for many reasons. Whether or not this is only true for games explicitly built for this purpose is less clear. Prensky (2001*a*) believes that all games have the potential to facilitate learning despite the intention of the creator. This could be due to the similarities observed between learning theory and game characteristics, discussed in Chapter 4, and applied to Drazah in Chapter 6. Because of this likeness, it is reasonable to expect some form of learning to take place in all games. However, the difficulty lies in how to target that learning to something beneficial rather than incidental learning of very specialised/peculiar learning of the game environment.

The work of Prensky appears to have been the starting point for much of the research in this area, and was often used to justify the initial assumption that games are potentially effective learning tools. Chapter 4 examines the existing work on increasing the effectiveness of educational games by applying theories such as Social Cognitive theory or including a pedagogical foundation such as experiential or guided learning theory. This research suggests there is a common agreement that the motivational and engaging properties of games help facilitate learning. However there did appear to be some conflict on whether harnessing this should be the main focus of an educational game.

Two differing motivations to game-based learning were presenting in Chapter 4 and are a possible cause behind differing methodologies in the field of game-based research:

1. The desire to harness the motivational power of games in order to 'make learning fun'.

2. A belief that 'learning through doing' in games such as simulations offers a powerful learning tool.

Although these motivations are not mutually exclusive, it is the belief of the author that an emphasis on learning, by adopting an existing learning approach such as experiential learning or an activity based interaction creates the most effective learning environment. This aligns with point 2 (above) as well as much of the work in the area of game-based learning and was the chosen approach for Drazah's learning (Chapter 6).

Throughout the project, it was noticed that elements of game design and learning theory often complimented each other. An example of this is the common pronouncement of the educational benefits of feedback in learning research and the fact that games almost always provide dynamic, salient feedback to the player. It is not unreasonable, then, to assume that

this parallel is likely to be one of the most fertile areas for developing effective game-based learning, and should be considered when implementing a design framework. Providing the usefulness of this similarity would be a great boon to educational games research, but is far beyond the scope of the work presented here.

Games have the ability to present a multitude of problems to the player in various environments, including simulations of real world conditions. This allows the player to conduct exploratory behaviour and helps expose new concepts and facilitate experiential learning. These ideas were incorporated into the design of Drazah, and are exhibited in a number of ways (Chapter 6). For example, the way in which cards present problems and potential solutions to a player creates an opportunity for experiential learning, particularly when combined with the feedback mechanism.

The use of a scoring system provides the player with feedback on their progress, and helps to enforce motivation and the construction of conceptual knowledge. It is intended that continual exposure through play will increase this knowledge to the point where it has an impact in real life. Literature on the motivational and engagement aspects of games ((Ryan et al. 2006), (Paras & Bizzocchi 2005) and (Nakamura & Csikszentmihalyi 1990), for example) highlighted techniques designed to encourage the replayability of a game, and thereby increase the likelihood of repeated exposure. Evidence of how these techniques were incorporated into the design of Drazah can be found in Chapter 6 and 7.

One of the more intriguing aspects of the project was discovering the multitude of physiological and psychological effects games have on players. In particular the way in which games seem to encourage motivation both intrinsically and extrinsically is fascinating and could be a key contributor to their success as entertainment products. Additionally, research on pedagogical foundations and learning theory was extremely fruitful, and

the impact of this can be seen in the conscious decision to base Drazah's design on experiential learning (Chapter 6).

Although empirical evaluation was not within the scope or remit of this project, the issues surrounding it have come up regularly in the literature. The problem is one mostly of complexity and scale: the complexity of the psychological and physiological (Chapter 3) effects on players and the scale of the studies that would be required to effectively factor out confounding issues (Chapter 5). While it appears reasonable to assume that game elements that are similar to concepts of learning theory indicate educational value in games, it is difficult to be sure that including them in games will yield the expected results. That is, is this just the fallacy of affirming the consequent? It seems clear that the field of educational games research needs to develop methods of empirical evaluation that can be used to determine the efficiency and efficacy of game-based learning and test the relationship between learning theory and certain game characteristics. Parallels between game design and learning theories suggest that it is possible to design educational games with guaranteed knowledge acquisition purely from an educational research perspective. Verifying this assumption is something that has yet to be performed.

8.1 Lessons Learnt

This project set out to highlight the parallels that exist between learning theory and game characteristics in order to determine if it is feasible to accentuate the characteristics of games that correspond in learning theory to facilitate effective learning through play. To do this, the physiological and psychological impact of games on players was considered, as well as the efficacy of educational games, and how they are designed. Once several educational game approaches were discussed (Chapter 4) the design of

an educational game could begin.

Drazah was produced as an exemplar of educational game design, and how certain game characteristics could be implemented to help facilitate learning. The development of Drazah's learning theory was inferred by the research on educational game design presented in Chapter 4, and serves to demonstrate the inclusion of various game characteristics and how they parallel learning theory.

The development of Drazah demonstrates how learning theory and game characteristics can be considered throughout the design of a game, and used to facilitate learning. Chapter 4 presents several pedagogical approaches used in games, and each have advantages and disadvantages to game-based learning. However it appears that, certain characteristics of the game itself, can have a significant impact on how effectively this pedagogy is deployed. Therefore, it is beneficial to consider how the pedagogy will be implemented, and design the game to deliver the style of user interaction and experience that best facilitates that learning style.

8.2 Future work

Although no formal evaluation of Drazah was conducted within this project, an analysis of methodologies used in the subject area was conducted and provides an indication of what a future testing phase may contain (Chapter 5). Findings suggest that an observational analysis is a common, and effective approach to determining levels of knowledge acquisition in game–based learning. However, due to the behavioural nature of the educational material in Drazah, it is unlikely any short-term evaluation would show any significant results. It is the believe of the author that a long period of testing, evaluating the knowledge acquisition levels of children playing the

game through pre-post test assessment could indicate potential learning. However determining whether or not this knowledge gain (if any) results in behavioural change would require longer periods of testing, like those found in the studies on behavioural education in Section 4.3.

Th effective use of a control group shown in the research of Rosas et al. (2002) had demonstrated the advantages of being able to compare the impact of game-based learning against normal delivery techniques. In turn, this also adds to the sample size required for an effective testing phase of Drazah, combined with the longevity of the testing period required to return significant data, it is suitable to say that the potential for worthy testing lie beyond the reach of this project, but are the aspirations of the author.

Appendix A

Protocol Syntax

A.1 SendStatus Expanded

$$
\begin{aligned}
\langle \text{HDeck} \rangle &\models \text{`}DECKH:\text{'} \langle \text{Hazard} \rangle \text{`,'} \langle \text{Cardlist} \rangle \\
\langle \text{Hazard} \rangle &\models \langle \text{<string>} \rangle \\
\langle \text{Cardlist} \rangle &\models \langle \text{`['{<Card>,}}^{4}\text{`]'} \rangle \\
\langle \text{Card} \rangle &\models \text{`}CARD:\text{'} \langle \text{<string>} \rangle \\
\langle \text{ADeck} \rangle &\models \text{`}DECKA:\text{'} \langle \text{Action} \rangle \text{`,'} \langle \text{<Cardlist>} \rangle \\
\langle \text{Action} \rangle &\models \langle \text{<string>} \rangle \\
\langle \text{Stack} \rangle &\models \text{`}STACK:[\text{'} \langle \text{{<Card>}`]'}^{3} \rangle \\
\langle \text{Score} \rangle &\models \text{`}SCORES:[\text{'} \langle \text{{<Float>}} \rangle \text{`]'} \\
\langle \text{State} \rangle &\models \text{`}STATE:\text{'} \langle \text{<string>} \rangle \\
\langle \text{Achievements} \rangle &\models \text{''}ACHIEVEMENTS:\text{''} \langle \text{<int>} \rangle \\
\langle \text{Wins} \rangle &\models \text{`}WINS:\text{'} \langle \text{<int>} \rangle \\
\langle \text{Losses} \rangle &\models \text{`}LOSSES:\text{'} \langle \text{<int>} \rangle
\end{aligned}
$$

$$\langle \text{Draws} \rangle \models \text{`}DRAWS:\text{'} \langle \text{<int>} \rangle$$

A.2 GameStart Communication

$$\begin{aligned} \langle \text{GameStart} \rangle \models\ & \langle \text{`MSG:'} \rangle \\ & \langle \text{HDeck} \rangle \\ & \langle \text{ADeck} \rangle \end{aligned}$$

A.3 Player Name

$$\langle \text{Name} \rangle \models \text{`}PLAYER:\text{'} \langle \text{<string>} \rangle \langle \text{newline} \rangle$$

A.4 GetInfo Communication

$$\begin{aligned} \langle \text{GetInfo} \rangle \models\ & \text{`}Opponent:\text{'} \langle \text{<string>} \rangle \langle \text{newline} \rangle \\ & \text{`}OWINS:\text{'} \langle \text{<int>} \rangle \langle \text{newline} \rangle \\ & \text{`}OLOSSES:\text{'} \langle \text{<int>} \rangle \langle \text{newline} \rangle \\ & \text{`}ODRAWS:\text{'} \langle \text{<int} \rangle \langle \text{newline} \rangle \\ & \text{`}OACHS:\text{'} \langle \text{<int>} \rangle \langle \text{newline} \rangle \end{aligned}$$

A.5 Achievements

⟨dbItems⟩ $\models$ ‘$BASE\_ACHIEVEMENTS:'$ ⟨<int>⟩⟨newline⟩

A.6 SendAll Communication

⟨SendAll⟩ $\models$ ‘$MSG : Player'$⟨<int>⟩‘$has\ played\ the\ situation\ card'$⟨<string>⟩⟨newline⟩

A.7 AskCard Communication

⟨AskCard⟩ $\models$ ‘$REQ : H'$⟨newline⟩

⟨AskCard⟩ $\models$ ‘$REQ : A'$⟨newline⟩

A.8 Debrief Communication

⟨Debrief⟩ $\models$ $`RDEBRIEF :' \; stack' : [' \langle$<string>$\rangle `],' \; scores' :' \; \langle$<string>$\rangle \langle$newline$\rangle$

A.9 Game Details Communication

⟨Details⟩ $\models$ $`GAMEDETAILS :' \; pDetails' : [' \langle$<string>$\rangle `]'$

$$(A.2)$$

A.10 Common Expressions

⟨String⟩ $\models$ ⟨{Char}⟩

⟨Char⟩ $\models$ $\{any \; character\}$

⟨Float⟩ $\models$ $\{< 0 \ldots 9 >\} `.' \{< 0 \ldots 9 >\}$

⟨Int⟩ $\models$ $\{0 \ldots 9\}$

⟨newline⟩ $\models$ $"\backslash n"$

Appendix B

Drazah client source code

B.1 card.py

```
import pygame

class Card(object):
    def __init__(self, shortTitle, title, code, img):
        self.shortTitle=shortTitle
        self.title=title
        self.code=code#self.img[:-4]
        self.img=img
        self.rImg=None
        self.rect=None

    def __str__(self):
        return self.__repr__()
    def __repr__(self):
        return 'Card("%s","%s","%s","%s")'%\
               (self.shortTitle,self.title,self.code,self.img)

    def loadImage(self):
        #print "!"*4,"LOADING: ",self.img
        cardFile = "images/"+self.img
        #print "OPENIGN %s"%cardFile
        newImage = pygame.image.load(cardFile).convert()
        oRect=newImage.get_rect()
        nw,nh=(200,250)
```

```
            if oRect.width>oRect.height: nw,nh=nh,nw
            newImage = pygame.transform.smoothscale\
                       (newImage,(nw,nh)).convert()
            self.rImg=newImage
            self.rect=self.rImg.get_rect()
            #screen.blit(newImage,(x,y))
            pass


        def setPos(self,x,y):
            self.rect.top=y
            self.rect.left=x
        def draw(self,surface):
            surface.blit(self.rImg,self.rect)


class Situation(Card):
    def __init__(self, shortTitle, title, code,img):
        Card.__init__(self,shortTitle,title, code,img)

class Hazard(Card):
    def __init__(self, shortTitle, title, code,img):
        Card.__init__(self,shortTitle,title, code,img)

class Action(Card):
    def __init__(self,shortTitle,title,code,img):
        Card.__init__(self,shortTitle,title, code,img)
```

B.2 client.py

```
#Imports go here:
import pygame.font, pygame.event, pygame.draw, string
import pygame,sys,socket,time, traceback
import sys
from pygame.locals import *
from deck import *
from threading import Thread
from Queue import Queue
from card import Card
from player import addAchievement, checkAchievement, \
    achievementList, Player

#Images to be loaded go here:
lScreen = pygame.image.load("images/drazah_lobby.png")
```

```
dScreen = pygame.image.load("images/drazah_draw.png")
wScreen = pygame.image.load("images/awaiting.png")
gui = pygame.image.load("images/drazah_bg.png")
goLogo = pygame.image.load("images/player_go.png")
goLogo = pygame.image.load("images/player_go.png")
rScreen = pygame.image.load("images/drazah_review.png")
dealingLogo = pygame.image.load("images/dealing.png")
eOGScreen = pygame.image.load("images/game_over.png")
playAgainButton = pygame.image.load("images/play_again.png")
quitButton = pygame.image.load("images/quit.png")

#achievement list
icons = []
for i in range (0,9):
    icons.append(pygame.image.load("images/A_%d.png"%i))

#waiting animation
awaiting_img = []
animcount = 0
for i in range (1,16):
    awaiting_img.append(pygame.image.load("images/waiting/%d.gif"%i))

achievements=0
newAchievementDisplay=[]
#For storing new achievements ready to display in debrief Class for
#connection thread


def drawBigAchievements(screen,nums):
    size=100
    for i in range(len(nums)):
        xpos=35+i*size
        ypos=435
        screen.blit(pygame.transform.smoothscale(icons[nums[i]],(size,size))\
                    .convert_alpha(),(xpos,ypos))

class ConnectionMonitor(Thread):

    def __init__(self, ip, port, incoming, outgoing):
        self.ip=ip
        self.port=port
        self.incoming=incoming
        self.outgoing=outgoing
        Thread.__init__(self)

    def run(self):
```

```
        sys.stdout.write("Running")
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((self.ip, self.port))
        s.setblocking(0)
        stack=""
        while 1:
            #Any data?
            try:
                data=s.recv(4096)
                stack+=data
            except:
                pass #no data
            if not self.outgoing.empty():
                data=self.outgoing.get()
                s.sendall(data)
            while stack.find("\n")!=-1:
                pos=stack.find("\n")
                data=stack[:pos]
                stack=stack[pos+1:]
                self.incoming.put(data.strip())
            time.sleep(1)

#Functions go here:
def get_key():
  while 1:
    pass

#Display text-entry box for player to enter name.
def display_box(screen, message):
  "Print a message in a box in the middle of the screen"
  screen.blit(lScreen,(0,0))
  if len(message) != 0:
    screen.blit(fontobject.render(message, 1, (0,0,0)),(50,150))
    pygame.display.flip()
#Display text box and run event loop for keyboard and mouse.
def ask(screen, question):
    "ask(screen, question) -> answer"
    pygame.font.init()
    current_string = []
    while 1:
        display_box(screen, question + ": " + \
                    string.join(current_string,""))
        inkey=None
        mousePos=None
        mousePressed=False
```

```
        event = pygame.event.poll()
        if event.type == KEYDOWN:
            inkey=event.key
        elif event.type == MOUSEBUTTONDOWN:
            mousePos=event.pos
            mousePressed=True

        if inkey!=None:
            if inkey == K_BACKSPACE:
                current_string = current_string[0:-1]

            elif inkey == K_RETURN:
                break
            elif inkey == K_MINUS:
                current_string.append("_")
            elif inkey <= 127:
                current_string.append(chr(inkey))
                display_box(screen, question + ": " + \
                            string.join(current_string,""))
        elif mousePos!=None:
            if mousePressed and startRect.collidepoint(mousePos):
                break
    return string.join(current_string,"")



#Quit function
def quitGame():
    conn.outgoing.put("END:QUIT")
    sys.exit(0)

#Lobby screen
w,h=size=(1024,768)
logo_prop=17.37

#Rectangles for each deck
hRect=pygame.Rect(10,508,400,250)
aRect=pygame.Rect(414,508,400,250)
stackRect=pygame.Rect(312,50,400,100)

#Rectangles for play again and quit buttons
pAgainRect=pygame.Rect(54,573,382,127)
quitRect=pygame.Rect(600,573,382,127)

#Rectangle for start game button
```

```
startRect=pygame.Rect(631,486,332,137)

#Set colour values
black=(0,0,0)
white=(255,255,255)
bg=(0,0,40)
fg=white

#Boolean for request highlighting (selection)
myTurn = False

#Initiate pygame and set screen size and font
pygame.init()
screen = pygame.display.set_mode(size)
pygame.display.set_caption('drazaH')
fontobject = pygame.font.Font(None,58)
pName=""
#Set loop to run 'ask' function until we have a player name
while pName=="": pName =  ask(screen, "Name")

#Set play to True and start running game
play=True
pygame.font.init()
sysfont=pygame.font.Font(None,12)
clock=pygame.time.Clock()

#Connect to Drazah server (after listening for address)
host = ""
port = 52133
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
s.bind((host,port))

#Run loop to listen for drazaH server address
while 1:
    try:
        message,address = s.recvfrom(8192)
        if message.startswith("DRAZAH:"):
            gameAddress,port = message[7:].strip().split(":")
            port=int(port)
            if len(gameAddress) > 0:
                break
        else:
            pass
    except (KeyboardInterrupt, SystemExit):
        raise
```

```
    except:
        traceback.print_exc()

#Start a connection to the game server
conn=ConnectionMonitor(gameAddress,port,Queue(),Queue())
conn.start()
log=[]
hazardDeck = None
actionDeck = None
stack = None
state = "AWAITING"
message=""
score = 0
time.sleep(2)

#True when in "round debrief mode"
inRDebrief=False
dbrfInfo=None
req=None
countdown=0.0
oName="no player"
rndScores = ""
dbrfScores = []
winner = None
#To turn off/on ingame display
ingameDisplay = True
while play:
    if state!="AWAITING": message=""
    ingameDisplay=True
    hoverCard=None
    screen.fill(bg)
    y=0
    while not conn.incoming.empty() and not inRDebrief:
        item=conn.incoming.get()

        #get players hazard deck from log
        if item.startswith("DECKH:"):
            hazardDeck=eval(item[6:])
            for c in hazardDeck.cards:
                c.loadImage()

        #Give player name to server
        elif item.startswith("REQ_NAME"):
            conn.outgoing.put("NAME:%s"%pName)

        #Get players action deck from log
```

```
elif item.startswith("DECKA:"):
    actionDeck=eval(item[6:])
    for c in actionDeck.cards:
        c.loadImage()

#Read log for current game stack, load images of cards in
#stack
elif item.startswith("STACK:"):
    stack=eval(item[6:])
    for stackItem in stack:
        stackItem.loadImage()

#Check the log for a state up date
elif item.startswith("STATE:"):
    newState=item[6:].strip()
    state = newState

#Read log for player score, update with new value
elif item.startswith("SCORE:"):
    newScore = item[6:].strip()
    score = newScore
    req=None

#Read log for a server request
elif item.startswith("REQ:"):
    req = item[4:].strip()
    myTurn = True

#Read Opponent info sent from game
elif item.startswith("OPPONENT:"):
    oName = item[9:].strip()
elif item.startswith("PLAYER:"):
    pName = item[7:].strip()

#Read info and set round debrief phase if necessary
elif item.startswith("RDEBRIEF:"):
    inRDebrief=True
    dbrfInfo=eval(item[9:])
    countdown= 1000.0
    req=None
    for i in  dbrfInfo["scores"]:
        dbrfScores.append(i)

#Set up base achievements so stuff pulled from DB don't get
#flagged as "new"
elif item.startswith("BASE_ACHIEVEMENTS:"):
```

```
            achievements=int(item[18:].strip())
        #Read info for new achievements gained
        elif item.startswith("ACHIEVEMENTS:"):
            newAchievments=int(item[13:].strip())
            if newAchievments!=achievements:
                added=achievements^newAchievments
                for i in achievementList(added):
                    print "Player just earned achievement %d"%i
                    newAchievementDisplay.append(i)
            achievements=newAchievments



        #Read log to get both player scores and determine who is the
        #winner
        elif item.startswith("GAMEDETAILS:"):
            pDetails = eval(item[12:])
            print "\n\n","-"*20,"\n",item[12:],"\n","-"*20,"\n\n"

            for i in pDetails["pDetails"]:
                for n in range (len(i)):
                    #Check both player scores to see who won
                    if i[0].score > i[1].score:
                        print "\n\nFIRST PLAYER WINS\n\n"
                        winner = i[0]
                        loser = i[1]
                    elif i[1].score > i[0].score:
                        print "\n\nSECOND PLAYER WINS\n\n"
                        winner = i[1]
                        loser = i[0]
                    elif i[0].score == i[1].score:
                        print "\n\nNOBODY WINS>\n\n"
                        winner = None
                        loser = None
                        screen.blit(dScreen,(0,0))

        #Eval data for side display
        elif item.startswith("WINS:"):
            displayWins = eval(item[5:])

        elif item.startswith("LOSSES:"):
            displayLosses = eval(item[7:])

        elif item.startswith("DRAWS:"):
            displayDraws = eval(item[6:])
```

```
        #Eval opponent data for side display
        elif item.startswith("OWINS:"):
            displayOWins = eval(item[6:])

        elif item.startswith("OLOSSES:"):
            displayOLosses = eval(item[8:])

        elif item.startswith("ODRAWS:"):
            displayODraws = eval(item[7:])

        elif item.startswith("OACHS:"):
            displayOAchs = eval(item[6:])

        #Reset command for it opponent drops out
        elif item.startswith("RESET:"):
            message = item[6:].strip()
            print "Opponent dropped: %s"%(message)
            hazardDeck = None
            actionDeck = None
            stack = None
            state = "AWAITING"
            #state = ""
            score = 0
            inRDebrief=False
            dbrfInfo=None
            req=None
            countdown=0.0
            pName="no player"
            oName="no player"
            rndScores = ""
            dbrfScores = []
            winner = None
            ingameDisplay = True
            newAchievementDisplay=[]
            break

        log.append(item)

    #Display 'Waiting' screen if state equal awaiting (turn off
    #ingameDisplay, player logo and dealing logo)
    if state == "AWAITING":
        if animcount < 15:
            screen.blit(awaiting_img[animcount],(0,0))
            animcount += 1
            time.sleep(0.3)
        else:
```

```
            animcount = 0

        if message!="":
            screen.blit(fontobject.render(message,1,\
                                          (0,0,0)),(280,110))
        pygame.display.flip()
        ingameDisplay = False
    # else:
    #     screen.blit(gui,(0,0))

    #Draw player turn logos if ingameDisplay equals true
    if ingameDisplay == True:
        screen.blit(gui,(0,0))
        #Draw the ingameDisplay details (players scores and ID's)
        #Display score
        font = pygame.font.Font(None, 45)
        text = font.render("%s"%score, 1, (10, 10, 10))
        textpos = text.get_rect()
        textpos =(945,30)
        screen.blit(text, textpos)
        #Display player ID
        font = pygame.font.Font(None, 36)
        text = font.render("%s"%pName, 1, (10, 10, 10))
        textpos = text.get_rect()
        textpos =(20,60)
        screen.blit(text, textpos)
        #Display opponent ID
        font = pygame.font.Font(None, 36)
        text = font.render("%s"%oName, 1, (10, 10, 10))
        textpos = text.get_rect()
        textpos =(20,95)
        screen.blit(text, textpos)

        #Player 0 stats (left side)
        #Display name
        font = pygame.font.Font(None, 18)
        text = font.render(pName, 1, (10, 10, 10))
        textpos = text.get_rect()
        textpos =(61,145)
        screen.blit(text, textpos)
        #Display Stats
        tWins = font.render("%s"%displayWins,1,(10,10,10))
        tLosses = font.render("%s"%displayLosses,1,(10,10,10))
        tDraws = font.render("%s"%displayDraws,1,(10,10,10))
        tWinspos = text.get_rect()
        tWinspos = (98,170)
```

```
tLossespos = text.get_rect()
tLossespos = (98,194)
tDrawspos = text.get_rect()
tDrawspos = (98,218)
screen.blit(tWins,tWinspos)
screen.blit(tLosses,tLossespos)
screen.blit(tDraws,tDrawspos)
#Display Player 0 achievements icons (left side)
startX = 20
startY = 245
size = 60
sizeY = 43
#Draw players achievements
for a in achievementList(achievements):
    xp = startX + (a%2)*size
    yp = startY + (a/2)*sizeY
    image = icons[a]
    image = pygame.transform.smoothscale\
            (image,(40,40)).convert_alpha()
    screen.blit(image,(xp,yp))


#Player 1 stats (right side)
#Display name
font = pygame.font.Font(None, 18)
text2 = font.render(oName, 1, (10, 10, 10))
text2pos = text.get_rect()
text2pos =(894,145)
screen.blit(text2, text2pos)
#Display Stats
tOWins = font.render("%s"%displayOWins,1,(10,10,10))
tOLosses = font.render("%s"%displayOLosses,1,(10,10,10))
tODraws = font.render("%s"%displayODraws,1,(10,10,10))
tOWinspos = text.get_rect()
tOWinspos = (898,170)
tOLossespos = text.get_rect()
tOLossespos = (898,194)
tODrawspos = text.get_rect()
tODrawspos = (898,218)
screen.blit(tOWins,tOWinspos)
screen.blit(tOLosses,tOLossespos)
screen.blit(tODraws,tODrawspos)
#Display Player 1 achievements icons (right side)
startX = 905
startY = 245
size = 60
```

```
        sizeY = 43
        #Draw players achievements
        for a in achievementList(displayOAchs):
            xp = startX + (a%2)*size
            yp = startY + (a/2)*sizeY
            image = icons[a]
            image = pygame.transform.smoothscale\
                    (image,(40,40)).convert_alpha()
            screen.blit(image,(xp,yp))

        #Draw logo for players turn
        if myTurn == True:
            screen.blit(goLogo,(230,55))
        elif myTurn == False:
            screen.blit(goLogo,(230,92))

    #Dealing logo
    if ingameDisplay == True:
        dealingLogo = pygame.transform.smoothscale\
                      (dealingLogo,(175,175)).convert_alpha()
        screen.blit(dealingLogo,(425,50))

#Draw players hazard cards from deck
    if hazardDeck!=None:
        #Draw hazard cards in slot
        ncards=len(hazardDeck)
        bw=hRect.width
        highlight=None
        for i in range(ncards):
            xpos=hRect.left+i*bw/ncards
            c=hazardDeck.cards[i]
            c.rect.left=xpos
            c.rect.top=515
            if req=="H" and \
                        not inRDebrief and \
                        highlight==None and \
                        c.rect.collidepoint\
                        (pygame.mouse.get_pos()) and \
                        myTurn == True:
                highlight=i
                hoverCard=c
            else:
                c.draw(screen)
        if highlight!=None:
            c=hazardDeck.cards[highlight]
            c.rect.move_ip(0,-50)
```

```
            c.draw(screen)

    #Draw players action cards from deck
    if actionDeck !=None:
        #Draw action cards in slot
        ncards=len(actionDeck)
        bw=aRect.width
        highlight = None
        for i in range(ncards):
            xpos=aRect.right-i*bw/ncards
            c=actionDeck.cards[i]
            c.rect.left=xpos
            c.rect.top=515
            if req=="A" and \
                    not inRDebrief and \
                    highlight == None and \
                    c.rect.collidepoint\
                    (pygame.mouse.get_pos()) and \
                    myTurn == True:
                highlight=i
                hoverCard=c
            else:
                c.draw(screen)
        if highlight!=None:
            c=actionDeck.cards[highlight]
            c.rect.move_ip(0,-50)
            c.draw(screen)

    #Draw round review screen
    if inRDebrief:
        screen.blit(rScreen,(0,0))
        drawBigAchievements(screen,newAchievementDisplay)
    #Draw stack
    if stack !=None:
        ncards=len(stack)
        bw=stackRect.width
        #If the game is NOT in debrief mode, draw the stack in
        #standard position
        if not inRDebrief:
            for i in range(ncards):
            #Draw the situation card above the others due to different
            #shape
                if i == 0:
                    xpos=stackRect.left+i*bw/ncards
                    c=stack[i]
                    c.rect.left=xpos+75
```

```
                c.rect.top=50
                c.draw(screen)
        #Other cards in stack that need to be drawn
            else:
                if i == 1:
                    xpos = stackRect.left+i*bw/ncards
                    c=stack[i]
                    c.rect.left= xpos-200
                    c.rect.top = 200
                    c.draw(screen)
                else:
                    xpos = stackRect.left+i*bw/ncards
                    c=stack[i]
                    c.rect.left= xpos
                    c.rect.top = 200
                    c.draw(screen)
    #If the game is in debrief mode, draw the stack in review
    #position
    else:
        for i in range(ncards):
            #Situation card in stack (review position)
            if i == 0:
                xpos= 50
                c=stack[i]
                c.rect.left=xpos
                c.rect.top=200+25
                c.draw(screen)
            else:
                #hazard card in stack (review position)
                if i == 1:
                    c=stack[i]
                    c.rect.left= xpos + 350
                    c.rect.top = 200
                    c.draw(screen)
                #Action card in stack (review position)
                else:
                    c=stack[i]
                    c.rect.left= xpos + 700
                    c.rect.top = 200
                    c.draw(screen)


#Draw end of game details (winner name & score and loser name &
#score)
if state == "END":
    #End of game page
```

```
        playAgainRect=playAgainButton.get_rect().move(54,573)
        quitRect=quitButton.get_rect().move(600,573)
        while 1:
            if winner!=None:
                screen.blit(eOGScreen,(0,0))
                #Display Winner name
                font = pygame.font.Font(None, 70)
                text = font.render("%s"%winner.name, 1, (10, 10, 10))
                textpos = text.get_rect()
                textpos =(60,190)
                screen.blit(text, textpos)
                #Display Winner score
                font = pygame.font.Font(None, 55)
                text = font.render("%s"%winner.score, 1, (10, 10, 10))
                textpos = text.get_rect()
                textpos =(309,347)
                screen.blit(text, textpos)
                #Display Loser name
                font = pygame.font.Font(None, 70)
                text = font.render("%s"%loser.name, 1, (10, 10, 10))
                textpos = text.get_rect()
                textpos =(630,190)
                screen.blit(text, textpos)
                #Display Loser score
                font = pygame.font.Font(None, 55)
                text = font.render("%s"%loser.score, 1, (10, 10, 10))
                textpos = text.get_rect()
                textpos =(880,347)
                screen.blit(text, textpos)
            else:
                print "\n\n␣DEBUG:01\n\n"
                print "I␣think␣the␣winner␣is:␣%s␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣(but␣this␣is␣where␣a␣draw␣should␣go!)"%winner
                #Loses the winner name (becomes None) until play again
                #button is pressed
                screen.blit(dScreen,(0,0))
            drawBigAchievements(screen,newAchievementDisplay)
            #Display play again and quit buttons
            screen.blit(playAgainButton, playAgainRect)
            screen.blit(quitButton,quitRect)
            mousePos=None
            event = pygame.event.poll()
            if event.type == MOUSEBUTTONDOWN:
                mousePos=event.pos
            if mousePos!=None:
                #process to restart a game using the 'play again'
```

```
                #button
                if playAgainRect.collidepoint(mousePos):
                    conn.outgoing.put("END:PLAY AGAIN")
                    state="AWAITING"
                    hazardDeck = None
                    actionDeck = None
                    stack = None
                    #state = ""
                    score = 0
                    inRDebrief=False
                    dbrfInfo=None
                    req=None
                    countdown=0.0
                    pName="no player"
                    oName="no player"
                    rndScores = ""
                    dbrfScores = []
                    #winner = None (falls in to no winner loop if left
                    #in)
                    ingameDisplay = True
                    newAchievementDisplay=[]
                    break
                elif quitRect.collidepoint(mousePos):
                    quitGame()
                    break
            pygame.display.flip()

    # Display Review screen pair scores
    if len(dbrfScores) > 0 and inRDebrief == True:
        font = pygame.font.Font(None, 50)
        text = font.render("+ %s" %dbrfScores[0], 1,(255,255,255))
        textpos = text.get_rect()
        textpos = (312,298)
        screen.blit(text,textpos)
        #Pair 2 (h/a)
        font = pygame.font.Font(None, 50)
        text2 = font.render("+ %s" %dbrfScores[1], 1,(255,255,255))
        textpos2 = text2.get_rect()
        textpos2 = (632,298)
        screen.blit(text2,textpos2)

    #Start event loop
    for event in pygame.event.get():
        if event.type == QUIT:
            play=False
        #Use the mouse position to get the players card selection, add
```

```
        #to stack.
        elif not inRDebrief and event.type == MOUSEBUTTONDOWN:
            if hoverCard != None:
                conn.outgoing.put("PLAY:%s"%hoverCard.code)
                myTurn = False
            else:
                myTurn = True

    #Use game clock to count down game review section (5 seconds)
    dt=clock.tick()
    if inRDebrief:
        countdown -= dt
        if countdown <= 0:
            inRDebrief = False
            #remove dbrfScores Item [0 and 1]
            dbrfScores = []
    pygame.display.flip()
```

B.3 controller.py

```
from server_send_and_receive_test import *
from game import *
from player import *
from time import sleep
import sys
from socket import *
import thread

#start connection
port=12345
broadport=52133
if len(sys.argv)>1:
    port=int(sys.argv[1])

pipe=ServerPipe("",port)
pipe.startConnection()

ip = "127.0.0.1" #"10.16.1.90"

#Broadcasting function so clients can find game
def broadcastServer(ip,broadport):


    s = socket(AF_INET, SOCK_DGRAM)
```

```
    s.bind((ip,broadport))
    s.setsockopt(SOL_SOCKET, SO_BROADCAST, 1)

    while 1:
        data = "DRAZAH: %s:%s"%(ip,port)
        s.sendto(data,("<broadcast>", broadport))
        time.sleep(2)

thread.start_new_thread(broadcastServer,(ip,broadport))

games=[]
playersWaiting=[]



while True:

    if pipe.connections.qsize()>0:
        con=pipe.connections.get()
        player=Player()
        player.setConnection(con)
        player.score=0
        playersWaiting.append(player)
    while len(playersWaiting)>=2:
        g=Game()

        g.stateStep()
        g.playerList.append(playersWaiting[0])
        g.playerList.append(playersWaiting[1])
        games.append(g)
        del playersWaiting[0:2]
    for g in games:
        err=False
        #print "0:",g.playerList[0].connection.fileno()
        #print "1:",g.playerList[1].connection.fileno()
        try:
            g.stateStep()
        except error:
            print "Player has disconnected"
            err=True
        #    g.playerList[0].connection
        if err:
            for i in g.playerList:
                try:
                    i.connection.sendall("")
                except error:
```

```
                    g.playerList.remove(i)

            for i in g.playerList:
                i.connection.sendall("RESET:\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣The␣other␣player␣disconnected.\n")
                i.newGameReset()
                playersWaiting.append(i)
                #g.playerList.remove(i)
            g.state="DELETE"
        #g.stateStep()
        playersWaiting.extend(g.recycle)
        g.recycle=[]
        if g.state=="DELETE":
            games.remove(g)
    sleep(0.5)
```

B.4 deck.py

```
#deck class
from card import Card
class Deck(object):
    def __init__(self,deckName,contents=None):
        self.cards=contents
        if contents==None: self.cards=[]
        self.deckName = deckName
    def __str__(self):
        return self.__repr__()
    def __repr__(self):
        #return "(%s) Deck contains (%s)"%(self.deckName,self.cards)
        return "Deck(\"%s\",%s)"%(self.deckName,self.cards)
    def append(self, card):
        self.cards.append(card)

    def __len__(self):
        return len(self.cards)

    def pop(self, index):
        """ pop the card (and return it) from the given index"""
        try:
            return self.cards.pop(index)
        except IndexError:
            print "SERIOUS␣ERROR."
            print "Asked␣to␣pop", index
            print "Len␣cards:", len(self.cards)
```

```
    def __contains__(self,card):
        #print "Deck asked to check %s"%card
        for i in self.cards:
         #   print "Is it %s?"%(i.code),
            if i.code==card:
          #      print "Yes!"
                return True
            else:
                pass
           #     print "No:("
        return False

    def remove (self,card):
        forRemoval = None
        for i in self.cards:
            if i.code==card:
                forRemoval=i
                break
        if forRemoval!=None:
            self.cards.remove(forRemoval)
            return forRemoval
        else:
            raise ValueError("Card cannot be removed - not in deck")


#create the object for a single deck, then call each object for
#S_deck, H_deck and A_deck

if __name__ == "__main__":
    c=Card("a","a card", "A/A1","A1.png")
    deck = Deck( "Hazard")
    deck.append(c)
    print deck
    d2=eval(deck.__repr__())
    print d2
```

B.5 game.py

```
from server_send_and_receive_test import *
from game import *
from player import *
from time import sleep
import time
```

```
import sys
import random
#SQL imports
from sqlalchemy import create_engine
from sqlalchemy import Table,Column,Integer,String,\
    MetaData,ForeignKey,Float
from sqlalchemy.orm import mapper
from sqlalchemy.orm import sessionmaker
from player import Player
#Class imports
from card import *
from deck import *
from player import *
from server_send_and_receive_test import *
from score import *

badwords={"fucker":"flipper","shit":"shiznit"}


#connect to player database
#create session
engine = create_engine\
        ("sqlite+pysqlite:///database.db",echo = True)
Session = sessionmaker(bind=engine)
session = Session()

metadata = MetaData()
#name table, set columns
drazah_players = Table("players",metadata,
                       Column("ID",Integer),
                       Column\
                       ("name",String,primary_key = True),
                       Column("wins",Integer),
                       Column("losses",Integer),
                       Column("draws",Integer),
                       Column("score",Float),
                       Column("achievements", Integer)
                       )

#Map class to db
mapper(Player,drazah_players)
Session.configure(bind=engine)


# ----------#
#Start Drazah
```

```
rounds=2
replayWindow=20

#main class
class Game(object):
    def __init__(self):

        #declaring pipe
        self.pipe = ServerPipe("",10001)

        #player list (empty list)
        self.playerList = []

        #player1(emptry list)
        self.ip = None
        self.port = None
        self.state = "NEW"
        self.buildDecks()
        self.stack=[]

        #Set up 2 connections
        self.scoreMatrixSH = ScoreMatrix\
                             ("../data/situation_hazard_pairs.csv")
        self.scoreMatrixHA = ScoreMatrix\
                             ("../data/hazard_action_pairs.csv")

        #When scores are calculated, drop them in here as a list of
        #two.  Gets used when sending round debrief information
        self.scoreTmp=None
        self.timeout=None
        self.recycle=[]
        self.id=random.randint(0,100000)
    #Set ip and port
    def setupConnection(self,ip,port):
        self.ip = ip
        self.port = port

    def startConn(self):
        self.pipe.startConnection(self.ip,self.port)
        self.pipe.recvData()

        #get 2 players check to see how many players are currently in
        #the game, if there is only 1 the game will add another, if
        #there is more than 2 players the game will inform the server
        #of this
```

```
    def printPlayers(self):
        print self.playerList
        #build decks Function that creates 3 instances of the 'Deck'
        #object and populates them with items

    def buildDecks(self):
        #build Hazard deck and populate with cards using the
        #information from 'hazarCardsContent.csv' file in the /data
        #folder. (open file >>set range (number of cards)>>read each
        #line >>append to list)
        self.H_deck=Deck("Hazard")
        f = open('../data/hazardCardsContent.csv')
        for i in range (1,27):
            line = f.readline().strip()
            items = line.split("#")
            h=Hazard(items[0],items[1],"H/"+items[2][:-4],items[2])
            self.H_deck.append(h)
        f.close()
        #build Action deck and populate with cards using the
        #information from 'actionCardsContent.csv' file in the /data
        #folder. (open file >>set range (number of cards)>>read each
        #line >>append to list
        self.A_deck=Deck("Action")
        f = open('../data/actionCardsContent.csv')
        for i in range (1,28):
            line = f.readline().strip()
            items = line.split('#')
            self.A_deck.append(Action(items[0],items[1],"A/"+\
                                      items[2][:-4],items[2]))
        #build Situation deck and populate with cards using the
        #information from 'situationCardsContent.csv' file in the
        #/data folder. (open file >>set range (number of cards)>>read
        #each line >>append to list
        self.S_deck=Deck("Situation")
        f = open('../data/situationCardsContent.csv')
        for i in range (1,rounds):
            line = f.readline().strip()
            items = line.split('#')
            self.S_deck.append(Situation(items[1],items[3],items[0][0]+"/"+\
                                         items[0][1],items[2]))
        # All .csv files were copied and tweaked from those used to create
        # the physical cards, card class attributes had to be cut down
        # because '#' was used to break up the descriptions for the cards,
        # however we do not need them here.

#allocate cards
```

```
    def popR(self, deck):
        try:
            s=random.randint(0,len(deck)-1)
            return deck.pop(s)
        except IndexError:
            raise Exception("Cannot popR - \
            There are no more cards in the deck!")

    #Deal cards
    def dealCards(self):
        #get cards from the H and A decks and put them in
        #the deck list inside playerList(Player)
        if len(self.playerList) == 2:
            for i in range(4):
                for p in self.playerList:
                    p.hDeck.append(self.popR(self.H_deck))
                    p.aDeck.append(self.popR(self.A_deck))
        else:
            raise Exception("A game cannot be started without two players")

    #set Sit card
    #function to play randomly selected situation card
    def pickSitCard(self):
        picked=self.popR(self.S_deck)
        return picked

    #function to add hazard card to players deck after they have
    #played one
    def addHazCard(self):
        pickedH = self.popR(self.H_deck)
        return pickedH

    #function to add action card to players deck after they have
    #played one
    def addActCard(self):
        pickedA = self.popR(self.A_deck)
        return pickedA

    #function to send messages to all players
    def sendAll(self,message):
        for p in self.playerList:
            try:
                p.connection.sendall(message+"\n\r")
            except:
                pass
    #Send status
```

```
def sendStatus(self):
    for p in self.playerList:
        try:
            p.connection.sendall("DECKH:"+str(p.hDeck)+"\n\r")
            p.connection.sendall("DECKA:"+str(p.aDeck)+"\n\r")
            p.connection.sendall("STACK:"+str(self.stack)+"\n\r")
            p.connection.sendall("SCORE:"+str(p.score)+"\n\r")
            p.connection.sendall("STATE:"+str(self.state)+"\n\r")
            p.connection.sendall("ACHIEVEMENTS:"+str(p.achievements)+\
                                 "\n\r")
            p.connection.sendall("WINS:"+str(p.wins)+"\n\r")
            p.connection.sendall("LOSSES:"+str(p.losses)+"\n\r")
            p.connection.sendall("DRAWS:"+str(p.draws)+"\n\r")
        except:
            p.connection.sendall("DECKH:"+str(p.hDeck)+"\n\r")
            p.connection.sendall("DECKA:"+str(p.aDeck)+"\n\r")
            p.connection.sendall("STACK:"+str(self.stack)+"\n\r")
            p.connection.sendall("SCORE:"+str(p.score)+"\n\r")
            p.connection.sendall("STATE:"+str(self.state)+"\n\r")
            p.connection.sendall("ACHIEVEMENTS:"+str(p.achievements)+\
                                 "\n\r")
            p.connection.sendall("DISPLAYINFO:"+str(p.wins)+str\
                                 (p.losses)+str(p.draws)+"\n\r")
            p.connection.sendall("WINS:"+str(p.wins)+"\n\r")
            p.connection.sendall("LOSSES:"+str(p.losses)+"\n\r")
            p.connection.sendall("DRAWS:"+str(p.draws)+"\n\r")
            pass
    pass

# STATE MACHINE
def stateStep(self):
    #print "State step, game:<%s> is in state:\t
    #%s"%(self.id,self.state) print "Current state: %s"%self.state
    data=""
    #Send client their name for display
    if self.state!="POSTGAME":
        for p in self.playerList:
            if p.name.strip()!="":
                #print "\tSending PLAYER info to",p.name
                p.connection.sendall("PLAYER:␣%s"%p.name+"\n\r")
    poinum="1"
    pnoinum="2"
    if self.state!="NEW" and self.state!="POSTGAME":
        #Player of interest
        poi=self.playerList[0]
        pnoi=self.playerList[1]
```

```
            if "2" in self.state:
                poi,pnoi=pnoi,poi
                poinum,pnoinum=pnoinum,poinum
        if self.state == "NEW":
            if len(self.playerList) == 2:
                self.dealCards()
                #print "Got two players."
                self.sendAll("MSG:The game begins")
                self.playerList[0].connection.sendall("%s\n\r"%\
                                                      self.playerList[0].hDeck)
                self.playerList[0].connection.sendall("%s\n\r"%\
                                                      self.playerList[0].aDeck)
                self.playerList[1].connection.sendall("%s\n\r"%\
                                                      self.playerList[1].hDeck)
                self.playerList[1].connection.sendall("%s\n\r"%\
                                                      self.playerList[1].aDeck)
                self.sendAll("REQ_NAME")
                self.state="GETINFO"
            else:
                pass
                #print "You need 2 players to start a game"
        if self.state=="GETINFO":
            if self.playerList[0].name!="" and \
               self.playerList[1].name!="":
                for p in self.playerList:
                    op=self.playerList[0]
                    if op==p: op=self.playerList[1]
                    #send opponent info to players
                    p.connection.sendall("OPPONENT: %s\n\r"%op.name)
                    p.connection.sendall("OWINS: %s \n\r"%op.wins)
                    p.connection.sendall("OLOSSES: %s \n\r"%op.losses)
                    p.connection.sendall("ODRAWS: %s \n\r"%op.draws)
                    p.connection.sendall("OACHS: %s \n\r"%op.achievements)
                self.state = "W_P1_S"
            else:
                for ii in range(len(self.playerList)):
                    i=self.playerList[ii]
                    #if i.name!="": continue
                    try:
                        data = i.connection.recv(4096)
                    except:
                        data=""
                    if data.startswith("NAME:"):
                        i.name=data[5:]
                        for j in badwords:
                            if i.name.find(j)>-1:
```

```
                            i.name=i.name.replace(j,badwords[j])
                        i.name=i.name[:10]
                        dbitems=session.query(Player).filter\
                                ("name=:name").params(name=i.name).all()
                        if len(dbitems)>0:
                            #Should only be 1 item.  Assuming so.  All
                            #hell will break loose if this is not the
                            #case.
                            tPlayer=dbitems[0]
                            tPlayer.connection=i.connection
                            tPlayer.hDeck=i.hDeck
                            tPlayer.aDeck=i.aDeck
                            tPlayer.score=0.0
                            tPlayer.connection.sendall("BASE_ACHIEVEMENTS:\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣%d\n"%tPlayer.achievements)
                            self.playerList[ii]=tPlayer
                        else:
                            session.add(i)
                            session.commit()

        elif self.state in ["W_P1_S","W_P2_S"]:
            if len(self.S_deck) > 0:
                #print "Waiting for player %s to play situation
                #card"%poinum send state to clients and wait for P1 to
                #choose card
                self.stack.append(self.pickSitCard())
                #print "Player %s has played card..."%poinum
                self.sendAll("MSG:Player␣%s␣has␣played␣the␣situation␣card␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣%s\n\r"%(poinum,str(self.stack[0])))
                self.sendStatus()
                pnoi.connection.sendall("REQ:H\n\r")
                #Send message to players
                self.state = "W_P%s_H"%pnoinum
            else:
                self.state = "END"


        elif self.state in ["W_P2_H","W_P1_H"]:
            #Check player 2's pipe to see if it's been played.  If it
            #has, change state Must be non-blocking
            try:
                data = poi.connection.recv(4096)
                if data.startswith("PLAY:"):
                    card = data[5:].strip()
                    #print card
                    if card in poi.hDeck:
```

```
                        #take card from player's deck
                        card=poi.hDeck.remove(card)
                        #give player a new card in return
                        poi.hDeck.append(self.addHazCard())
                        #Add it to the pile
                        self.stack.append(card)
                        #print self.stack Send out new status (using
                        #the function we wrote)
                        self.sendStatus()
                        # Send REQ:A to player 1
                        pnoi.connection.sendall("REQ:A\n\r")
                        #Set state to W_P1_A
                        self.state = "W_P%s_A"%pnoinum
                    else:
                        print "Card not in player's deck"

                else:
                    print "While in state %s, got odd data from client: \
                    %s"%(self.state,data)
            except:
                #print "No data while in state: %s"%self.state
                sleep(0.5)

#NEW STATE#

        elif self.state in ["W_P1_A", "W_P2_A"]:
            #check pipe...
            try:
                data = poi.connection.recv(4096)
                if data.startswith("PLAY:"):
                    card = data[5:].strip()

                    if card in poi.aDeck:
                        #remove card from players hand
                        card=poi.aDeck.remove(card)
                        #give player a new card in return
                        poi.aDeck.append(self.addActCard())
                        self.stack.append(card)
                        #print self.stack
                        #Calculate s/h score add to poi
                        s = self.stack[0]
                        h = self.stack[1]
                        score=self.scoreMatrixSH.score(s,h)
                        self.scoreTmp=[score]
                        pnoi.score += score
                        #Check achievement 0
```

```
                    if score>=10 and not pnoi.checkAchievement(0):
                        pnoi.addAchievement(0)
                    # pnoi.connection.sendall("ACHGAINED: 0 \n\r")
                    #print "Hand score:",score
                    #Add this to the player's score
                    poi.score + score
                    #Calculate H/A score for pnoi
                    a = self.stack[2]
                    score = self.scoreMatrixHA.score(h,a)
                    #Check achievement 1
                    if score>=10 and poi.checkAchievement(1) !=1:
                        poi.addAchievement(1)
                        # poi.connection.sendall("ACHGAINED: 1 \n\r")
                    #Check achievement 6
                    if pnoi.score > poi.score and self.stack[1].code=="W*"\
                            and pnoi.checkAchievement(6) !=1:
                        pnoi.addAchievement(6)
                        # pnoi.connection.sendall("ACHGAINED: 6 \n\r")
                    #Check achievement 7
                    if pnoi.score > poi.score and self.stack[1].code=="H*" \
                            and pnoi.checkAchievement(7) !=1:
                        pnoi.addAchievement(7)
                        # pnoi.connection.sendall("ACHGAINED: 7 \n\r")
                    #Add scores
                    self.scoreTmp.append(score)
                    #Add this to player's score
                    poi.score += score
                    #print "Hand score:",score
                    #print "Player 1's current score %d"%\
                    #(self.playerList[0].score)
                    #print "Player 2's current score %d"%\
                    #(self.playerList[1].score)
                    #self.sendAll("STATUS: W_P2_S, %s"%str(self.stack))
                    self.sendStatus()
                self.state = "RDB:W_P%s_S"%pnoinum
            except:
                #print "No data while in state: %s"%self.state
                sleep(0.5)

        elif self.state.startswith("RDB:"):
            #Round debrief
            self.sendStatus()
            self.sendAll("RDEBRIEF:{'stack':[%s],␣'scores':%s}\
␣␣␣␣␣␣␣␣␣␣␣␣"%(self.stack,self.scoreTmp))
            #Clear stack
            self.stack=[]
```

```
            self.state=self.state[4:]
            if self.playerList[0].score>self.playerList[1].score:
                self.playerList[0].wins+=1
                self.playerList[1].losses+=1
            elif self.playerList[1].score>self.playerList[0].score:
                self.playerList[1].wins+=1
                self.playerList[0].losses+=1
            else:
                for i in self.playerList:
                    i.draws+=1
            session.commit()

        elif self.state == "END":
            #print "End of game!"

            for p in [[self.playerList[0],self.playerList[1]],
                      [self.playerList[1],self.playerList[0]]]:
                #Check achievement 2
                if p[0].wins == 1 and p[0].checkAchievement(2) != 1:
                    p[0].addAchievement(2)
                #Check achievement 3
                if p[0].wins>=10 and p[0].checkAchievement(3)!=1:
                    p[0].addAchievement(3)
                #Check achievement 4
                if p[0].wins>=50 and p[0].checkAchievement(4) !=1:
                    p[0].addAchievement(4)
                #Check achievement 5
                if p[0].wins > 5 and p[0].wins >= p[0].losses*2 \
                        and p[0].checkAchievement(5)!=1:
                    p[0].addAchievement(5)
                #Check achievement 8
                if p[0].score > 30 and p[0].checkAchievement(8)!=1:
                    p[0].addAchievement(8)
                #Check achievement 9
                if p[0].score > 60 and p[0].checkAchievement(9)!=1:
                    p[0].addAchievement(9)
            #Send status and game details to clients

            out="GAMEDETAILS:{'pDetails':[%s]}"%(self.playerList)
            self.sendStatus()
            #print "Im sending: ",out
            self.sendAll(out)
            self.sendStatus()
            self.timeout=time.time()
            self.state="POSTGAME"
        elif self.state=="POSTGAME":
```

```
            for i in self.playerList:
                try:
                    data = i.connection.recv(4096)
                except:
                    data=""

                if data.startswith("END:"):
                    request = data[4:].strip()
                    #print request
                    if request == "PLAY AGAIN":
                        #print "Player wants to start a new game"
                        i.newGameReset()
                        self.recycle.append(i)
                        self.playerList.remove(i)
                    elif request == "QUIT":
                        i.connection.close()
                        #print "closing connections"
                        self.playerList.remove(i)
            if time.time()-self.timeout>replayWindow:
                for i in self.playerList:
                    i.connection.close()
                self.playerList=[]
            if len(self.playerList)==0 :
                self.state="DELETE"
```

B.6 pipe.py

```
import socket, time
import sys

class Pipe(object):
    def __init__(self,ip, port,socket):
        self.ip = ip
        self.port = port
        self.socket = socket
        #self.socket = socket.socket(socket.AF_INET,
        #socket.SOCK_STREAM)

    def startConnection(self):
        #Port we wish to serve on
        SERVER_PORT=5000

        #create STREAMing socket (TCP)
        serversocket = socket.socket\
```

```
                (socket.AF_INET, socket.SOCK_STREAM)

#Bind to any/default interface on the local machine
serversocket.bind(("", SERVER_PORT))

#become a server socket
serversocket.listen(5)


#List of clients - fill this up as we go
clients=[]


#Set blocking to OFF.  Without threads, this is the only way
#we can deal with multiple users.  If blocking is ON, we will
#be stuck at the accept() funciton call until someone connects
serversocket.setblocking(0)

while True:
    try:
        #accept connections from outside. Note we get a new
        #socket object, leaving the original one free for more
        #users to connect
        (clientsocket, address) = serversocket.accept()
        #add it to the client list
        clients.append(clientsocket)
        #Set blocking to OFF.  Same as before, but this time
        #we don't want to block on the recv() call
        clientsocket.setblocking(0)
    except:
        #Non-blocking sockets throw an exception if asked to
        #connect/recieve when there's nothing to do. They do
        #this rather than block.  We need to cope with that,
        #so we catch the exception and move on
        pass


    #Keep a list of messages ready to be sent
    messages=[]


    #See which clients have something to say
    for c in clients:
        try:
            #Catch the exception associated with non-blocking
            #sockets again
```

```
                data=c.recv(1024)
                if data!="" and data != None:
                    #If there is some data, put it into the queue
                    messages.append(data)
            except:
                pass

        #Now we have a list of messages sent from clients since
        #the last time through the loop, we send each item to
        #every client
        for c in clients:
            for m in messages:
                c.send(m)
        time.sleep(1)



if __name__ == "__main__":
    ip = ""
    port = 1234
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    test = Pipe(ip,port,sock)
    #test.startConnection()
```

B.7 pipe server.py

```
import socket, time
import sys

class Pipe(object):
    def __init__(self,ip, port,socket):
        self.ip = ip
        self.port = port
        self.socket = socket
        #self.socket = socket.socket(socket.AF_INET,
        #socket.SOCK_STREAM)

    def startConnection(self):
        #Port we wish to serve on
        SERVER_PORT=5000

        #create STREAMing socket (TCP)
        serversocket = socket.socket\
                    (socket.AF_INET, socket.SOCK_STREAM)
```

```
#Bind to any/default interface on the local machine
serversocket.bind(("", SERVER_PORT))

#become a server socket
serversocket.listen(5)


#List of clients - fill this up as we go
clients=[]


#Set blocking to OFF.  Without threads, this is the only way
#we can deal with multiple users.  If blocking is ON, we will
#be stuck at the accept() funciton call until someone connects
serversocket.setblocking(0)

while True:
    try:
        #accept connections from outside. Note we get a new
        #socket object, leaving the original one free for more
        #users to connect
        (clientsocket, address) = serversocket.accept()
        #add it to the client list
        clients.append(clientsocket)
        #Set blocking to OFF.  Same as before, but this time
        #we don't want to block on the recv() call
        clientsocket.setblocking(0)
    except:
        #Non-blocking sockets throw an exception if asked to
        #connect/recieve when there's nothing to do.  They do
        #this rather than block.  We need to cope with that,
        #so we catch the exception and move on
        pass


    #Keep a list of messages ready to be sent
    messages=[]


    #See which clients have something to say
    for c in clients:
        try:
            #Catch the exception associated with non-blocking
            #sockets again
            data=c.recv(1024)
```

```
                if data!="" and data != None:
                    #If there is some data, put it into the queue
                    messages.append(data)
            except:
                pass

        #Now we have a list of messages sent from clients since
        #the last time through the loop, we send each item to
        #every client
        for c in clients:
            for m in messages:
                c.send(m)
        time.sleep(1)



if __name__ == "__main__":
    ip = ""
    port = 1234
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    test = Pipe(ip,port,sock)
    #test.startConnection()
```

B.8 player.py

```
from deck import Deck
from random import randint
class Player(object):
    def __init__(self, name="", ID=None, score=0.0, wins=0,\
                 losses=0,draws=0,achievements=0):
        self.name = name
        # if self.name==None:
        #     self.name="Guest%d"%randint(1,2000)

        self.ID = ID
        if self.ID==None:
            self.ID=randint(1,200000000)
        self.score = score
        self.hDeck = Deck("Hazard")
        self.aDeck = Deck("Action")
        self.connection=None
        self.wins=wins
        self.losses=losses
        self.draws=draws
```

```
        self.achievements=achievements
    def newGameReset(self):
        self.score = 0
        self.hDeck = Deck("Hazard")
        self.aDeck = Deck("Action")
    def setConnection(self, con):
        self.connection=con

    def __str__(self):
        return self.__repr__()
    def __repr__(self):
        print "Score is:",self.score
        return "Player('%s',%d,%f,%d,%d,%d,%d)"%\
               (self.name, self.ID, self.score,self.wins,\
                self.losses,self.draws,self.achievements)

    def addWins(self):
        self.wins+=1

    def addLosses(self):
        self.losses+=1

    def addDraws(self):
        self.draws+=1

    def showStats(self):
        return "My name is %s (%d) and my score is %f\n\tWins: \
        %d\n\tLosses: %d\n\tDraws: %d\n\tCheevies:%s"%\
    (self.name, self.ID, self.score, self.wins, self.losses, \
     self.draws,repr(self.achievementList()))

    def addAchievement(self,chNum):
        self.achievements=addAchievement(self.achievements,chNum)

    def checkAchievement(self,chNum):
        return checkAchievement(self.achievements,chNum)

    def achievementList(self):
        return achievementList(self.achievements)


def addAchievement(mask,chNum):
    return mask | 2**chNum

def checkAchievement(mask,chNum):
    return (mask & 2**chNum)!=0
```

```
def achievementList(mask):
    out=[]
    for i in range(10):
        if mask & (2)**(i):
            out.append(i)
    return out



if __name__ == "__main__":
    players = [Player("Mike",8008135),Player("James",1337)]
    players[1].addAchievement(3)
    players[1].addAchievement(9)
    players.append(eval(repr(players[1])))

    for i in players:
        print i
        print i.showStats()
```

B.9 score.py

```
from card import *

class ScoreMatrix(object):
    def __init__(self,scorefile):
        self.scorefile=scorefile
        self.matrix={}
        f=open(scorefile)
        for l in f:
            items=l[:-1].split(",")
            self.matrix[(items[0],items[1])]=float(items[2])
        f.close()

    def score(self,card1,card2):
        h,o=card1,card2
        if h.__class__ !=Hazard:
            h,o=o,h
        print h
        print o
        if h.__class__!=Hazard or (o.__class__!=Situation and \
                                   o.__class__!=Action):
```

```
            raise TypeError("Score␣only␣works␣for␣Hazard/Action␣\
␣␣␣␣␣␣␣␣␣␣␣␣or␣Hazard/Situation␣pairs")
        try:
            pairScore=self.matrix[(o.code,h.code)]
        except KeyError:
            for i in self.matrix:
                print "\t",i
            raise KeyError("%s␣does␣not␣contain␣data␣for␣the␣pair␣\
␣␣␣␣␣␣␣␣␣␣␣␣[%s]␣-␣␣[%s]"%(self.scorefile,o.code,h.code))
        return pairScore

# if __name__=="__main__":
#     a=ScoreMatrix("../data/situation_hazard_pairs.csv")
#     s=Situation("bath","In a bath","S/2")

#     h=Hazard("Duck","Big duck","H/W1","duck.png","A challenger has appeared")
#     print a.score(s,h)
```

B.10 server send and receive test.py

```
import socket,time
import sys
from threading import Thread
from Queue import Queue


class ConnectionMonitor(Thread):

    def __init__(self, ip, port, connections):
        self.ip=ip
        self.port=port
        self.connections=connections
        Thread.__init__(self)

    def run(self):

        sys.stdout.write("Running")
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.bind((self.ip,self.port))
        s.listen(1)
        while 1:

            conn, addr = s.accept()
            print "connected␣to␣client␣@", addr
```

```
            self.connections.put(conn)
            conn.setblocking(0)

class ServerPipe(object):
    def __init__(self,ip,port):
       self.ip = ip
       self.port = port
       self.connections=Queue()

    def startConnection(self):
        sys.stdout.write("starting")
        self.cm=ConnectionMonitor(self.ip,self.port,\
                                  self.connections)
        self.cm.start()
        sys.stdout.write("Connection started")

    def closeConnection(self):
        self.conn.close()

    def requestData(self, item):
        self.conn.send(item)

if __name__ == "__main__":
    ip = "127.0.0.1"
    port = 1234
    if len(sys.argv)>1:
        port=int(sys.argv[1])
    sys.stdout.write("Beginning")
    sys.stdout.flush()
    p=ServerPipe(ip,port)
    sys.stdout.write("Starting outside")
    p.startConnection()
    while True:
        sys.stdout.flush()
        print "Connections:"
        for i in p.connections.queue:
            print i
        print data


        time.sleep(2)
```

B.11 broadcast.py

```
import sys, time
from socket import *

MYPORT = 52134

address = "127.0.0.1"


def broadcastServer(ip,port):
    s = socket(AF_INET, SOCK_DGRAM)
    s.bind((address,0))
    s.setsockopt(SOL_SOCKET, SO_BROADCAST, 1)

    while 1:
        data = "DRAZAH: %s"%address
        s.sendto(data,("<broadcast>", MYPORT))
        print data
        time.sleep(2)


broadcastServer(address,MYPORT)
```

Bibliography

Amory, A., Naicker, K., Vincent, J. & Adams, C. (1999), 'The use of computer games as an educational tool: identification of appropriate game types and elements.', *British Journal of Educational Technology* .

Amory, A. & Seagram, R. (2003), 'Educational game models: Conceptualisation and evaluation', *Journal of Higher Education* .

Atkins, P., Wood, R. & Rutgers, P. (2002), 'The effects of feedback format on dynamic decision making', *Organizational Behavior and Human Decision Processes* **88**(2), 587–604.

Baranowski, T., Buday, R., Thompson, D. I. & Baranowski, J. (2008), 'Playing for real: videogames and stories for health-related behaviour change', *American journal of preventive medicine* .

Barendregt, W., Bekker, M. & Speerstra, M. (2003), Empirical evaluation of usability and fun in computer games for children, *in* 'Proceedings of Human-Computer Interaction INTERACT-03', Vol. 3, pp. 705–708.

Becker, K. (2005), 'How are games educational? learning theories embodied in games', *DiGRA, Vancouver, Canada* .

Bellotti, F., Berta, R., Gloria, A. D. & Primavera, L. (2009), 'Enhancing the educational value of videogames', *ACM Computers in Entertainment* .

Brodzki, E. (n.d), Statistical Analysis of Gamer Behavior, PhD thesis, WORCESTER POLYTECHNIC INSTITUTE.

Burke, V., Mori, T., Giangiulio, N., Gillam, H., Beilin, L., Houghton, S., Cutt, H., Mansour, J. & Wilson, A. (2002), 'An innovative program for changing health behaviours', *Asia Pacific Journal of Clinical Nutrition* **11**, S586–S597.

Chuang, T.-Y. & Chen, W.-F. (2009), 'Effect of computer-based video games on children: an experimental study', *Educational Technology & Society, v12 n2 p1-10 2009* .

Conati, C. & Zhao, X. (2004), Building and evaluating an intelligent pedagogical agent to improve the effectiveness of an educational game, *in* 'Proceedings of the 9th international conference on Intelligent user interfaces', ACM, pp. 6–13.

Draganski, B., Gaser, C., Busch, V., Schuierer, G., Bogdahn, U. & May, A. (2004), 'Neuroplasticity: changes in grey matter induced by training', *Nature* **427**(6972), 311–312.

Eagle, M. (2009), Level up: a frame work for the design and evaluation of educational games, *in* 'Proceedings of the 4th International Conference on Foundations of Digital Games', ACM, pp. 339–341.

Economics, O. (2008), The economic contribution of the uk games development industry, Technical report, Oxford Economics.

ESA (2011), Essential facts about the computer and video games industry, Technical report, ESA.

Fairclough, C. & Cunningham, P. (2003), A multiplayer case based story engine., *in* 'GAME-ON'03', pp. 41–41.

Foundation, T. P. (1990), 'Object orientated programming'. `www.python.org`.

Fu, F., Su, R. & Yu, S. (2009), 'Egameflow: A scale to measure learners' enjoyment of e-learning games', *Computers & Education* **52**(1), 101–112.

Funk, J. B. (1993), 'Reevaluating the impact of videogames', *Clinical Pediatrics* .

Garzotto, F. (2007), Investigating the educational effectiveness of multiplayer online games for children, *in* 'Proceedings of the 6th international conference on Interaction design and children', ACM, pp. 29–36.

Gentile, D. A. (n.d.), Video games affect on the brain - for better or worse (unpublished). N/A.

Gneezy, U. & Rustichini, A. (2000), 'A fine is a price', *The Journal of Legal Studies* pp. 1,17.

Griffiths, M. (1999), 'Violent video games and aggression:: A review of the literature', *Aggression and violent behavior* **4**(2), 203–212.

Gros, B. (2007), 'digital games in education: The design of games-based learning environments', *Journal of research on technology in education* .

Harris, J. (2001), *The Effects of Computer Games on Young Children: A Review of Research*, Vol. 72, Home Office.

Herz, J. (2001), Gaming the system: What higher education can learn from multiplayer online worlds, *in* 'The Internet and the University: Forum', pp. 169–291.

Just Kid Inc. (2002), An environmental scan of children's interactive media from 2000-2002 (unpublished). 1.

Kafai, Y. B. (2006), 'playing and making games for learning: Instructionist and constructionist perspectives for game studies', *Games and Culture* .

Katsikopoulos, K. V. (2010), 'Psychological heuristics for making inferences: Denition, performance, and the emerging theory and practice', *Inform* **8**(1).

Kebritchi, M. & Hirumi, A. (2008), 'Examining the pedagogical foundations of modern educational computer games.', *Computers & Education* .

Kiili, K. (2004), 'Digital game based learning: Towards an experiential gaming model', *The Internet and higher education* .

Kirriemuir, J. & McFarlane, C. A. (2004), Literature review in games and learning, Technical report, FutureLab.

Leonard, D. (2004), 'Unsettling the military entertainment complex: videogames and a pedagogy of peace.', *SIMILE: Studies in Media & Information Literacy Education* .

Martin, R. (1998), 'Uml tutorial: Finite state machines', *Engineering Notebook Column, C++ Report* .

McLean, J. & Ernest, J. (1998), 'The role of statistical significance testing in educational research', *Research in the Schools* **5**(2), 15–22.

Nakamura, J. & Csikszentmihalyi, M. (1990), *Flow - The psychology of optimal experience*, Harper and Row, chapter 'The concept of flow' Chapter 7.

Paras, B. & Bizzocchi, J. (2005), Game, motivation, and effective learning: An integrated model for educational game design, *in* 'Proceedings of DIGRA', Citeseer.

Pattis, R. (n.d.), 'Ebnf: A notation to describe syntax'. http://www.cs.cmu.edu/ pattis/misc/ebnf.pdf.

Prensky, M. (2001*a*), *Digital Game-Based Learning*, Paragon House, chapter 'Digital game-based revolution:' Chapter 1.

Prensky, M. (2001*b*), *Digital game-based learning*, Paragon House, chapter 'The games Generations:' Chapter 2.

Prensky, M. (2002), 'What kids learn that's positive from playing video games', *Retrieved May 2010* .

Prensky, M. (2003), 'Digital game-based learning', *ACM computers in entertainment* .

pyGame (2000). www.pygame.org.

Rosas, R., Nussbaum, M., Cumsille, P., Marianov, V., Correa, M., Lopez, P., Rodrigeuz, P. & Salinas, M. (2002), 'Beyond nintendo: design and assessment of educational videogames for first and second grade students', *Computers and Education* .

RoSPA (2002), Home and leisure accident surveillance survey, Technical report, Royal Society for the Prevention of Accidents.

Roy, V., Dessai, S. & Yadav, S. (2009), 'Design and development of arm processor based web server', *International Journal of Recent Trends in Engineering* **1**(4), 94–98.

Ryan, R. & Deci, E. (2000), 'Intrinsic and extrinsic motivations: Classic definitions and new directions* 1', *Contemporary educational psychology* **25**(1), 54–67.

Ryan, R., Rigby, C. & Przybylski, A. (2006), 'The motivational pull of video games: A self-determination theory approach', *Motivation and Emotion* **30**(4), 344–360.

Shapley, L. (1953), 'Stochastic games', *Proceedings of the National Academy of Sciences of the United States of America* **39**(10), 1095.

Siwek, S. (2007), 'Video games in the 21st century', *Entertainment Software Association* **36**.

Smith, J. (2004), Playing dirty-understanding conflicts in multiplayer games, *in* '5th Annual Conference of The Association of Internet Researchers, The University of Sussex', Vol. 19, Citeseer.

SQLALchemy (2010). `www.sqlalchemy.org`.

Squire, K. (2003), 'Video games in education', *Int. J. Intell. Games & Simulation* .

Squire, K. (2005), 'Changing the game: What happens when videogames enter the classroom', *Journal of online education* .

Star, J. R. (2000), 'On the relationship between knowing and doing in procedural learning', *Proceedings of ICLS 2000: the University of Michigan, Ann Arbor, Michigan, USA, June 14th-17th, 2000* .

Statistics, N. (2002), Mortality statistics: Injury and poisoning england and wales, Technical report, National Statistics.

Sweetser, P. & Wyeth, P. (2005), 'Gameflow: a model for evaluating player enjoyment in games', *Computers in Entertainment (CIE)* **3**(3), 3–3.

van Bergen, W. (2011), 'Why developers should be force fed state machines'. `http://www.shopify.com/technology/3383012-why-developers-should-be-force-f`

Wiklund, M. (2005), Behavioural changes in students participating in an upper secondary education program using unmodified computer games as the primary teaching tool, *in* 'Proceedings of CGAMES 2005, 7: th International Conference on Computer Games, 28-

30 November 2005', Wolverhampton: University of Wolverhampton, School of Computing and Information Technologies.

Zea, N. P., Sanchez, J. L. G., Gutierrez, F. L., Cabrera, M. J. & P.Paderewski (2009), 'Design of educational multiplayer videogames: A vision from collaborative learning', *Advances in Engineering software* .